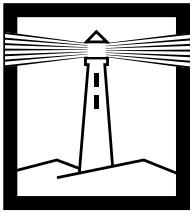


LOOKOUT™

Lookout™ Reference Manual

NATIONAL INSTRUMENTS™

March 1998 Edition
Part Number 321254B-01

Internet Support
E-mail: support@natinst.com
FTP Site: ftp.natinst.com
Web Address: http://www.natinst.com

Bulletin Board Support
BBS United States: 512 794 5422
BBS United Kingdom: 01635 551422
BBS France: 01 48 65 15 59

Fax-on-Demand Support
512 418 1111

Telephone Support (USA)
Tel: 512 795 8248
Fax: 512 794 5678

International Offices
Australia 03 9879 5166, Austria 0662 45 79 90 0, Belgium 02 757 00 20, Brazil 011 288 3336,
Canada (Ontario) 905 785 0085, Canada (Québec) 514 694 8521, Denmark 45 76 26 00, Finland 09 725 725 11,
France 01 48 14 24 24, Germany 089 741 31 30, Hong Kong 2645 3186, Israel 03 6120092, Italy 02 413091,
Japan 03 5472 2970, Korea 02 596 7456, Mexico 5 520 2635, Netherlands 0348 433466, Norway 32 84 84 00,
Singapore 2265886, Spain 91 640 0085, Sweden 08 730 49 70, Switzerland 056 200 51 51, Taiwan 02 377 1200,
United Kingdom 01635 523545

National Instruments Corporate Headquarters
6504 Bridge Point Parkway Austin, Texas 78730-5039 USA Tel: 512 794 0100

© Copyright 1996, 1998 National Instruments Corporation. All rights reserved.

Important Information

Warranty

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this manual is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of National Instruments Corporation.

Trademarks

Lookout™ and NI-DAQ™ are trademarks of National Instruments Corporation.

Product and company names listed are trademarks or trade names of their respective companies.

WARNING

Reliability of operation of the Lookout software can be impaired by adverse factors, including but not limited to fluctuations in electrical power supply, computer hardware malfunctions, computer operating system software fitness, fitness of compilers and development software used to develop a Lookout application, installation errors, software and hardware compatibility problems, malfunctions or failures of process monitoring or process control devices, transient failure of electronic systems (hardware and/or software), unanticipated uses or misuses, or error on the part of the user or application designer. (Adverse factors such as these are hereafter collectively termed "system failures.")

Any process or sub-part of a process where a system failure would create a risk of harm to property or persons (including the risk of bodily injury and death) should not be reliant solely upon electronic monitoring due to the risk of system failure. To avoid damage, injury or death, the user or application designer must take reasonably prudent steps to protect against system failures, including by not limited to the safeguard of independent stand-alone alarm mechanisms and the installation within easy reach of plant floor personnel of manual shutoff switches on dangerous equipment.

Contents

About This Manual

Organization of the Product User Manual xxxi
Conventions Used in This Manual xxxiii
Related Documentation xxxiv
Customer Communication xxxiv

Part I
Getting Started

Chapter 1
Installing Lookout

Hardware Requirements 1-1
Software Requirements 1-1
Installing Lookout 1-2
    To Install Lookout 1-2
Starting Lookout for the First Time 1-3
    To Start Lookout for the First Time 1-4
    Automatic Process Loading 1-5

Chapter 2
Introduction

Architecture 2-2
    What is an Object? 2-2
        Functionality 2-3
        Parameters 2-3
        Database 2-4
    Data Members 2-5
        Logical Data Members 2-5
        Numeric Data Members 2-6
        Text Data Members 2-10
        (implicit) Data Members 2-11
    Object Classes 2-11
    Object Connections 2-12
    Supervisory Control 2-12
    Event-Driven Processing 2-13
        Advantages of Active Notification 2-14
    Environment Services 2-15

Serial Port Communication Service 2-15
Database Service 2-16
Graphics Service 2-16
Alarm Service 2-16
Multimedia Service 2-16
Security Service 2-16
Historical Logging Service 2-17
ODBC Service 2-17
DDE Service 2-17
Networking Service 2-17
Redundancy Service 2-17

Chapter 3
Getting Started

Starting Lookout 3-1
To Start Lookout 3-1
To Open a Process File 3-2
The Lookout Screen 3-2
The Title Bar 3-2
The Menu Bar 3-3
The Status Bar 3-3
The Lookout Workspace 3-3
Control Panels 3-4
Operator Input 3-5
Virtual Keypad 3-5
Virtual Keyboard 3-5
The Development Environment 3-6
The Process File 3-6
The Source Code File 3-6
The State File 3-7
The Development Process 3-7
Configuration Shortcuts 3-8
Mouse Shortcuts 3-8
Remembering Tagnames 3-9

Chapter 4
Using Lookout

Selecting Objects 4-1
Creating Objects 4-2
To Create an Object 4-2
Object Tagnames 4-4
Editing Object Databases 4-5

To Edit Database Parameters....4-5
Numeric Member Parameters....4-8
Logical Member Parameters....4-12
Text Member Parameters....4-13
Importing and Exporting Object Databases....4-13
Exporting an Object Database....4-13
Creating a Database Spreadsheet....4-15
Importing an Object Database....4-16
Copying an Object Database....4-19
Connecting Objects....4-19
Connecting Data Members to Parameters....4-19
Identifying Object Data Members....4-20
Connecting Data Members to Data Members....4-21
Displaying Data Members on Control Panels....4-24
To Display a Data Member on a Control Panel....4-24

Chapter 5
Developer Tour

Building a Lookout Monitoring System....5-1
Create the Control Panel....5-1
Water Level Simulator....5-3
Adding Data Display....5-9
Adding Control Objects....5-11
Completing the Interface Panel....5-17
Connect the PLC and RTU....5-20
Conclusion....5-24

Part II
Lookout Features and Services

Chapter 6
Serial Communications

Introduction to Driver Objects....6-1
Understanding the Communications Service....6-2
Defining Serial Port Settings....6-3
Selecting the Serial Port....6-3
Setting Receive Gap....6-4
Selecting the Serial Connection....6-4
Hardwired Settings....6-4
RTS/CTS Handshaking Settings....6-4
Dial-Up Modem Settings....6-6

Chapter 7
Expressions

Creating Expressions 7-3
Expressions on Control Panels 7-3
Expression Objects 7-4
Expressions as Parameters 7-5
Expressions as Connections 7-6
Expression Dialog Box 7-8
Expression Syntax 7-9
White Space 7-9
Arithmetic Operators 7-10
Text Operator 7-10
Comparison Operators 7-11
Expression Functions 7-13
Logical Functions 7-14
Lookup Functions 7-16
Mathematical Functions 7-17
Statistical Functions 7-20
Text Functions 7-22
Trigonometric Functions 7-25
Date/Time Functions 7-27

Chapter 8
Graphics

Static Graphics 8-1
Displaying Text, Plates, Insets, Rectangles, and Lines 8-1
Displaying Static Custom Graphics 8-3
Dynamic Graphics 8-6
Displaying Dynamic Logical Signals 8-7
Displaying Dynamic Numeric Signals 8-9
Displaying Dynamic Text Signals 8-12
Creating Custom Graphics 8-13
Step-by-Step Example: Creating Custom Graphics 8-13
Creating the Graphic 8-14
Exporting the Graphic to Lookout 8-15
Testing the Graphic in Lookout 8-15
Graphic File Types 8-17
Bitmaps 8-17
Metafiles 8-17
Bitmaps or Metafiles? 8-17
Memory Considerations 8-18

Chapter 9
Alarms

Defining Alarm Conditions 9-1
Database-Generated Alarms 9-1
Alarm Objects 9-2
The Alarm Subsystem 9-3
Alarm Groups 9-3
Alarm Priorities 9-4
Alarms Window 9-5
Alarm Display Options 9-7
Alarm Filters 9-8
Alarm Print 9-10
Alarm Acknowledgment 9-11

Chapter 10
Security

Accounts 10-1
Process File Security 10-3
Control Security 10-4
Viewing Security 10-5
Control Panels 10-5
Controllable Objects 10-6
System Settings 10-6
Action Verification 10-8

Chapter 11
Logging Data and Events

Spreadsheet Logger 11-1
Data Location 11-2
CSV Files 11-3
File and Disk Errors 11-4
Concurrent File Access 11-4
Information Overload 11-5
Citadel Historical Database Logger 11-5
Data Location 11-6
Creating a Historical Database 11-6
Logging Criteria 11-8
Information Overload 11-8
Event Logger 11-9
Data Location 11-9
Information Overload 11-10

Report Generation 11-10
Control Panel Reports 11-10
Third-Party Reports 11-12

Chapter 12
Structured Query Language

Introduction 12-1
What is ODBC? 12-1
What is SQL? 12-1
Configuring the Citadel ODBC Driver 12-1
Accessing Citadel Data 12-3
Traces Table 12-3
Data Transforms 12-4
SQL Examples 12-5
Accessing Citadel Data with Microsoft Query 12-6
Accessing Citadel Data with Microsoft Excel 12-9
Accessing Citadel Data with Microsoft Access 12-10
Accessing Citadel Data with Visual Basic 12-12

Chapter 13
Dynamic Data Exchange

Linking Lookout to Other Applications 13-2
DDE Server 13-2
DDE Client 13-3
DDE Peer-to-Peer 13-4
DDE Alarms 13-6

Chapter 14
Networking

Networking Considerations 14-2
Multilink Networking 14-2
Linking Controllable Objects 14-3
Linking Controllable Objects Together Across a Network 14-3
Linking Non-Controllable Objects 14-5
To Access Real-Time Data at Another Computer 14-5
Table Networking 14-6
Hardware Networking 14-10
Multilink and Table Networking Comparison 14-11
Networking with DDE 14-11
Running NETDDE.EXE Automatically 14-12
Windows for Workgroups 14-12

Windows 95 14-12
Windows NT 14-12
Adding a Trusted DDE Share 14-13

Chapter 15
Redundancy

Standby Basics 15-2
Failover Scenarios 15-3
Configuring Standby 15-3
To Define Network Settings 15-4
To Enable File Sharing in Windows 3.1.1 15-5
To Enable File Sharing in Windows 95 and Windows NT 15-7
To Configure Standby Options 15-8

Chapter 16
Runtime Menu Commands

File Commands 16-1
File»New 16-1
File»Open 16-1
File»Close 16-2
File»Save 16-3
File»Reopen 16-3
File»Print»Alarms
File»Print»Events 16-4
File»Log on 16-4
File»Log off 16-5
File»Password 16-5
File»Exit 16-5
Edit Command 16-6
Edit»Edit Mode 16-6
Option Commands 16-6
Options»System 16-6
Options»Accounts 16-8
Options»Modbus 16-9
Options»Serial Ports 16-9
Options»Import APT Database 16-10
Run Commands 16-10
Run»Add 16-10
Run»Revise 16-11
Run»Delete 16-11
Alarm Commands 16-12
Alarms»Show 16-12

Alarms»Display Options 16-12
Alarms»Filter Options 16-12
Alarms»Print 16-13
Alarms»Select All 16-14
Alarms»Deselect All 16-14
Alarms»Acknowledge 16-14
Window Commands 16-14
Window»Arrange Icons 16-14
Window»Minimize All 16-14
Window»*nTitle* *16-15*
Window»More Windows 16-15

Chapter 17
Edit Mode Menu Commands

Edit Commands 17-1
Edit»Undo 17-1
Edit»Cut 17-2
Edit»Copy 17-2
Edit»Paste 17-2
Edit»Delete 17-3
Edit»Select All 17-3
Edit»Edit Mode 17-3
Insert Commands 17-4
Insert»Displayable Object 17-4
Insert»Expression 17-5
Insert»Text/Plate/Inset 17-5
Insert»Graphic 17-6
Insert»Scale 17-7
Insert»Control Panel 17-8
Insert»Positions 17-9
Object Commands 17-11
Object»Create 17-11
Object»Modify 17-13
Object»Delete 17-15
Object»Edit Connections 17-16
Object»Edit Database 17-16
Arrange Commands 17-16
Arrange»Align 17-17
Arrange»Space Evenly 17-17
Arrange»Group 17-18
Arrange»Ungroup 17-18
Arrange»Move to Front 17-18

Arrange»Move to Back 17-18
Change Functions 17-19
Change»Font 17-19
Change»Text Color 17-19
Change»Background Color 17-20
Change»Numeric Format 17-20
Change»Justify Text 17-21

Part III
Object Reference

Chapter 18
Object Class Reference

AB_PLC2,
AB_PLC5,
AB_SLC500 18-2
Allen-Bradley Serial Port Interface Parameters 18-4
Allen-Bradley DH+ Interface Parameters 18-5
Allen-Bradley Ethernet Interface Parameters 18-8
Using the 5136-SD card from S-S Technologies, Inc. 18-9
Allen-Bradley Data Members 18-9
Allen-Bradley Error Messages 18-16
Accumulator 18-22
Accumulator Data Member 18-23
AdvantechPCL 18-24
AdvantechPCL Data Members 18-25
Alarm 18-27
Alarm Data Members 18-29
$Alarm 18-30
Alarm Data Members 18-31
Using $Alarm with Other Objects 18-32
Alternator 18-33
Alternator Data Members 18-34
Connecting the Alternator 18-35
Command and Advance 18-35
Maximum Run Time and Delay Operation 18-36
Hand - Off - Auto Modes 18-36
Elapsed Time, Run Time, and Reset 18-37
Alternator Status Messages 18-37
Animator 18-39
Animations 18-40
Color Animation 18-41

Animator Data Members ... 18-43
Applicom ... 18-44
Applicom Data Members ... 18-46
General Information on Using the Applicom drivers for Lookout ... 18-61
Applicom Local and Image Modes ... 18-62
Configuration of the Applicom Server ... 18-63
Loading of the Applicom Server ... 18-63
Testing the Applicom Server ... 18-63
Creating the Cyclic Functions ... 18-64
Special Instructions on Using the Applicom Local Object Class ... 18-64
Applicom Status Messages ... 18-65
Aquatrol ... 18-67
RTU Configuration Dialog Box ... 18-68
Aquatrol Data Members ... 18-69
Aquatrol Status Messages ... 18-70
ASCII ... 18-72
ASCII Data Members ... 18-73
Request and Response Format Strings ... 18-75
ACSII Object Markers ... 18-76
Entering ASCII Object Format String ... 18-78
Request Frame Construction Examples ... 18-79
Response Format Examples ... 18-79
Using Sum Data Members ... 18-80
ASCII Error Messages ... 18-80
Average ... 18-82
Average Data Members ... 18-82
Counter ... 18-84
Counter Data Members ... 18-84
Cutler-Hammer ... 18-85
Cutler-Hammer Data Members ... 18-87
Cutler-Hammer Status Messages ... 18-88
DataTable ... 18-90
Multiplexing Displays and Graphics ... 18-91
DataTable Example ... 18-92
Connecting Signals to DataTables ... 18-94
The Display Panel ... 18-96
Operating Your Multiplexed Panel ... 18-97
DataTable Cursors ... 18-98
Using Multiple Cursors ... 18-99
DataTable Data Members ... 18-100
DdeLink ... 18-103
DdeLinks on Same Computer ... 18-103
DdeLinks to Remote Computer ... 18-103

DDELink Data Members....18-104
DdeTable....18-105
DdeTable on Same Computer....18-105
DdeTable to Remote Computer....18-107
DDETable Data Members....18-108
DelayOff....18-109
DelayOff Data Members....18-110
DelayOn....18-111
DelayOn Data Members....18-112
DeltaTau....18-113
DeltaTau Data members....18-113
Derivative....18-115
Derivative Data Members....18-116
DialGauge....18-117
DialGauge Data Members....18-119
DL205, DL405....18-120
DL205 and DL405 Data Members....18-123
DL205 and DL405 Status Messages....18-125
Dynamic....18-127
Dynamic Data Members....18-129
ElapsedTime....18-133
ElapsedTime Data Members....18-133
Event....18-134
Event Data Members....18-134
Expression....18-135
FisherROC....18-137
FisherROC Data Members....18-139
FisherROC Status Messages....18-142
Flipflop....18-144
FlipFlop Data Members....18-144
Gauge....18-145
Gauge Data Members....18-146
GE_Series6....18-147
GE_Series6 Data Members....18-149
GE_Series6 Status Messages....18-150
GE_Series90....18-152
GE_Series90 Data Members....18-153
GE_Series90 Status Messages....18-155
Histogram....18-157
Histogram Data Members....18-159
Hitachi....18-161
Hitachi Data Members....18-162

Hitachi Status Messages 18-164
HyperTrend 18-165
HyperTrend Data Members 18-171
Integral 18-173
Integral Data Members 18-174
Interpolate 18-175
Interpolate Data Members 18-178
Interval 18-179
Interval Data Members 18-180
IPASCII 18-181
IPASCII Data Members 18-182
Request and Response Format Strings 18-183
Markers 18-184
Entering the Format String 18-187
Request Frame Construction Examples 18-187
Response Format Examples 18-187
IPASCII Error Messages 18-188
Junction 18-190
Junction Data Members 18-190
$Keyboard 18-191
$Keyboard Data Members 18-192
LatchGate 18-194
LatchGate Data Members 18-194
Maximum 18-195
Maximum Data Members 18-195
Minimum 18-197
Minimum Data Members 18-197
Mitsubishi
MitsubishiFX 18-199
Mitsubishi Data Members 18-201
Mitsubishi Status Messages 18-202
Mitsubishi Models Supported 18-203
Modbus
ModbusMOSCAD 18-204
Advanced Modbus Parameters 18-207
Modbus Protocol Statistics 18-208
Modbus Data Members 18-209
ModbusMOSCAD Data Members 18-214
ModbusSlave 18-216
Modbus Slave Data Members 18-217
Multistate 18-219
Multistate Data Members 18-220

National Instruments Fieldbus 18-221
Fieldbus Alarms 18-223
Fieldbus Data Members 18-223
Fieldbus Status Messages 18-224
Fieldbus Troubleshooting 18-225
National Instruments FieldPoint 18-226
FieldPoint Data Members 18-228
FieldPoint Multiple Discrete Data Members 18-229
FieldPoint Error Messages 18-230
National Instruments Lookout OPC Client Driver 18-233
OPC Client Data Member Tables 18-234
OPC Item IDs in Lookout 18-235
OPC Item ID Format in Lookout 18-236
Examples 18-236
Adding a Single Alias 18-236
Browsing OPC Server Address Space 18-238
Importing Alias Lists 18-238
Neutralzone 18-240
NeutralZone Data Members 18-241
NIDAQDevice 18-242
NIDAQDevice Data Members 18-243
NIDAQ.INI 18-243
NIDAQDevice Error Messages 18-244
NISCXI 18-246
NISCXI Data Members 18-247
Configuring NIDAQ.INI for NISCXI 18-248
Channel Attributes 18-248
Cold-Junction Sensor Attributes 18-249
NISCXI Error Messages 18-249
SCXI Devices 18-250
Omron 18-251
Omron Data Members 18-253
Omron Status Messages 18-254
Omron Models Supported 18-254
OneShot 18-255
OneShot Data Members 18-256
Optomux 18-257
Optomux Watchdog Capability 18-259
Optomux Data Members 18-259
Optomux Status Messages 18-261
Pager 18-263
Pager Data Members 18-265
Pager Object Modes 18-265

Numeric Only 18-265
Alphanumeric Mode 18-265
Pager Queueing 18-266
Pager Status Messages 18-266
Panel 18-267
Manipulating Panels 18-270
Panel Switching 18-270
Special Considerations for “Home Panel” 18-272
Panel Print 18-273
Screen Resolution and Lookout Graphics 18-274
Panel Data Members 18-274
Pareto 18-276
Weighted or Unweighted Charts 18-278
Incrementing Factor Counts 18-279
Pareto Data Members 18-279
Philips 18-282
Philips Data Members 18-284
Philips Status Messages 18-285
Philips Alarms 18-285
PID 18-286
PID Positional Control 18-289
PID Velocity Control 18-289
PID Data Members 18-290
Pipe 18-292
Pipe Data Members 18-292
Playwave 18-293
Playwave Data Members 18-293
Pot 18-294
Pot Data Members 18-296
Profibus DP 18-297
Configuring the Profibus DP Network 18-297
Profibus DP Requirements 18-297
PFB Card Settings 18-298
Profibus DP Data Members 18-299
Profibus DP Status Messages 18-299
ProfibusL2 18-301
Lookout Messaging System 18-301
Sample Program 18-301
Detailed Explanation of the Profibus Example Program 18-302
DB1 Configuration 18-302
Function Block Explanation 18-303
ProfibusL2 Requirements 18-304
PFB Card Settings 18-305

ProfibusL2 Data Members 18-306
ProfibusL2 Status Messages 18-307
Pulse 18-309
Pulse Data Members 18-310
Pushbutton 18-311
Pushbutton Data Members 18-313
RKC F Series 18-314
RKC Data Members 18-316
RKC Status Messages 18-322
Recipe 18-324
Recipe Data Members 18-329
Reliance 18-331
PC-Link Card Settings 18-332
Destination Settings 18-332
Reliance Data Members 18-332
Reliance Data Members 18-333
Reliance Status Messages 18-333
Run 18-334
S5_3964 18-336
S5_3964 Data Members 18-337
S5_3964 Alarms 18-340
Sample 18-342
Sample Data members 18-342
SampleText 18-344
SampleText Data Members 18-344
Scale 18-345
Scale Data Members 18-346
SiemensTI505 18-347
Configuring HI-TF 18-348
SiemensTI505 Data Members 18-350
SiemensTI505 Status Messages 18-354
Sixnet 18-356
Sixnet Data Members 18-357
Importing Sixtags Database 18-360
Sixnet Status Messages 18-360
Spinner 18-362
Spreadsheet 18-364
Spreadsheet Data Members 18-367
SqlExec 18-368
SqlExec Data Members 18-369
SqlExec Comments 18-369
SQL Command Buffering 18-371
SqlExec Status Messages 18-372

SquareD 18-374
Serial Port Interface Parameters 18-375
SY/LINK Interface Parameters 18-375
SY/ENET Interface Parameters 18-377
SquareD Data Members 18-378
SquareD Error Messages 18-380
Switch 18-381
Switch Data Members 18-383
$System 18-384
System Data Members 18-384
Tesco 18-385
Tesco Data Members 18-386
TextEntry 18-389
TextEntry Data Members 18-391
TimeOf*xxxx* *18-392*
Timeofxxxx Data Members 18-393
Tiway 18-394
Communication Techniques 18-395
Local Port 18-395
Unilink Host Adapter 18-395
Unilink PC Adapter 18-396
CTI TCP/IP 18-398
Tiway Data Members 18-398
Importing APT Tag Files 18-402
Toshiba Mseries/Toshiba Tseries 18-403
Toshiba Data Members 18-405
Toshiba Status Messages 18-407
Trend 18-408
Trend Data Members 18-412
Wizdom 18-413
Wizdom Data Members 18-413
XBarR 18-415
XBarR Data Members 18-418
XChart 18-421
XChart Data Members 18-422
XYChart 18-424
XYChart Data Members 18-425

Appendix A
Customer Communication

Index

Figures

Figure 2-1. An object encapsulates data, parameters, and functionality in one bundle. 2-2
Figure 2-2. Switch Definition Dialog Box 2-4
Figure 2-3. Example connections between two objects 2-12
Figure 2-4. Example of conventional, Loop-Driven software program 2-13
Figure 2-5. Lookout object-oriented and event-driven architecture 2-14

Figure 3-1. The Expression Editor Activated by Right-clicking on a Yellow Data Field 3-10

Figure 4-1. The Select Object Class list box 4-2
Figure 4-2. An Object Definition Dialog Box 4-3
Figure 4-3. Pot Display Dialog Box 4-4
Figure 4-4. Alarm processing of a data member whose alarm setpoints are shown and whose **Deadband** is 12 4-11

Figure 6-1. Serial Port Settings Dialog Box 6-3

Figure 7-1. Example of **logical** Expression and Corresponding Display Dialog Box 7-4
Figure 7-2. Example of **numeric** Expression and Corresponding Display Dialog Box 7-4
Figure 7-3. Example of **text** Expression and Corresponding Display Dialog Box 7-4
Figure 7-4. Edit Connections Dialog Box 7-7
Figure 7-5. Insert Expression Dialog Box 7-8

Figure 8-1. Pot Object as a Slider, a Digital Entry, and Increment and Decrement Buttons 8-10
Figure 8-2. Using Expressions To Display Dynamic Text Signals 8-12

Figure 9-1. Creating Database-Generated Alarms 9-2
Figure 9-2. Creating an Alarm Object 9-3
Figure 9-3. Activating the Alarms Window from the Status Bar 9-5
Figure 9-4. Example Alarms Window 9-7
Figure 9-5. **Alarm Filters** Dialog Box 9-8
Figure 9-6. Determining the Number of Filtered Alarms from the Status Bar 9-9
Figure 9-7. Acknowledging Alarms 9-12

Figure 10-1. Create, Revise, or Delete System User Accounts in the **Account Maintenance** Dialog Box 10-2
Figure 10-2. Defining Security in the **System Options** Dialog Box 10-7

Figure 11-1. A Spreadsheet Object with Complex Mechanisms Defined for Triggering Logging 11-2
Figure 11-2. System Options Dialog Box Specifies Spreadsheet File Root Directory Path 11-3
Figure 11-3. Using the **PLC Database** Dialog Box to Create a New Trace 11-7
Figure 11-4. Printing a Normal Panel Object 11-11

Figure 12-1. The Query Window with `Traces` Table 12-7
Figure 12-2. The **Select Data Source** Dialog Box 12-11

Figure 14-1. Multilink Networking 14-3
Figure 14-2. Table Networking 14-7

Figure 18-1. Allen-Bradley Parameter Dialog Box 18-3
Figure 18-2. AB_SLC-500 Definition Parameters Dialog Box Configured for DH+ Communications 18-5
Figure 18-3. AB_PLC5 Definition Parameters Dialog Box Configured for Ethernet Communications 18-8
Figure 18-4. AB_PLC5 Definition Parameters Dialog Box Configured for the 5136-SD card 18-9
Figure 18-5. Accumulator Definition Parameters Dialog Box 18-22
Figure 18-6. AdvantechPCL Definition Parameters Dialog Box 18-24
Figure 18-7. Typical Settings for a Logical Style Alarm 18-27
Figure 18-8. Typical Settings for a Numeric Style Alarm 18-28
Figure 18-9. Edit Connections Dialog Box for using $Alarm 18-30
Figure 18-10. Alternator Definition Parameters Dialog Box 18-33
Figure 18-11. Select Graphic Dialog Box 18-39
Figure 18-12. Animator Definition Parameters Dialog Box, Animation Tab 18-40
Figure 18-13. Animator Definition Parameters Dialog Box, Color Tab 18-42
Figure 18-14. Applicom Definition Parameters Dialog Boxes 18-44
Figure 18-15. Aquatrol Definition Parameters Dialog Box 18-67
Figure 18-16. RTU Configuration Dialog Box 18-68
Figure 18-17. ASCII Definition Parameters Dialog Box 18-72
Figure 18-18. Average Definition Parameters Dialog Box 18-82
Figure 18-19. Counter Definition Parameters Dialog Box 18-84
Figure 18-20. Cutler-Hammer Definition Parameters Dialog Box 18-85
Figure 18-21. DataTable Definition Parameters Dialog Box 18-90
Figure 18-22. Graphical Representation of a DataTable Showing Connections 18-91
Figure 18-23. Edit Connections Dialog Box 18-94
Figure 18-24. Edit Connections Dialog Box 18-95
Figure 18-25. DataTable with Cursor at Row 2 and Corresponding Outputs 18-99
Figure 18-26. DataTable with Cursor at Row 9 and Corresponding Outputs 18-99
Figure 18-27. DdeLink Definition Parameters Dialog Box (Same Computer) 18-103

Figure 18-28. DdeLink Definition Parameters Dialog Box (Remote Computer) 18-104
Figure 18-29. DdeTable Definition Parameters Dialog Box (Same Computer) 18-105
Figure 18-30. Inserting a DdeTable Expression 18-106
Figure 18-31. DdeTable Definition Parameters Dialog Box (Remote Computer) 18-107
Figure 18-32. DelayOff Definition Parameters Dialog Box 18-109
Figure 18-33. DelayOff Display Parameters Dialog Box 18-110
Figure 18-34. DelayOn Definition Parameters Dialog Box 18-111
Figure 18-35. DelayOn Display Parameters Dialog Box 18-112
Figure 18-36. Derivative Definition Parameters Dialog Box 18-115
Figure 18-37. DialGauge Definition Parameters Dialog Box 18-117
Figure 18-38. DialGauge Display Parameters Dialog Box 18-118
Figure 18-39. DL205 Parameters Configured for one PLC in a Multidropped Configuration 18-121
Figure 18-40. Dynamic Parameter Configuration Dialog Box 18-128
Figure 18-41. ElapsedTime Definition Parameters Dialog Box 18-133
Figure 18-42. Typical Settings for an Event 18-134
Figure 18-43. Create Expression Dialog Box 18-135
Figure 18-44. FisherROC Definition Parameters Dialog Box 18-137
Figure 18-45. Flipflop Definition Parameters Dialog Box 18-144
Figure 18-46. Gauge Definition Parameters Dialog Box 18-145
Figure 18-47. Gauge Display Parameters Dialog Box 18-146
Figure 18-48. GE_Series6 Definition Parameters Dialog Box 18-147
Figure 18-49. GE_Series90 Definition Parameters Dialog Box 18-152
Figure 18-50. Histogram Definition Parameters Dialog Box 18-157
Figure 18-51. Hitachi Definition Parameters Dialog Box 18-161
Figure 18-52. HyperTrend Cursor Dialog Box 18-167
Figure 18-53. HyperTrend Dialog Box 18-168
Figure 18-54. Plot of a Logical Value 18-170
Figure 18-55. HyperTrend Display Parameters Dialog Box 18-171
Figure 18-56. Integral Definition Parameters Dialog Box 18-173
Figure 18-57. Interpolate Definition Parameters Dialog Box 18-175
Figure 18-58. Interval Definition Parameters Dialog Box 18-179
Figure 18-59. Interval Display Parameters Dialog Box 18-180
Figure 18-60. Junction Definition Parameters Dialog Box 18-190
Figure 18-61. Edit Connections Dialog Box 18-191
Figure 18-62. LatchGate Definition Parameters Dialog Box 18-194
Figure 18-63. Maximum Configuration Parameters Dialog Box 18-195
Figure 18-64. Minimum Configuration Parameters Dialog Box 18-197
Figure 18-65. Mitsubishi Configuration Parameters Dialog Box 18-199
Figure 18-66. Modbus Configuration Parameters Dialog Box 18-205
Figure 18-67. Advanced Modbus Parameters Dialog Box 18-207
Figure 18-68. Modbus Protocol Statistics Dialog Box 18-209
Figure 18-69. Modbus Slave Configuration Parameters Dialog Box 18-217

Figure 18-70. Multistate Configuration Parameters Dialog Box 18-219
Figure 18-71. Fieldbus Configuration Parameters Dialog Box 18-222
Figure 18-72. Fieldbus Alarms Dialog Box 18-223
Figure 18-73. National Instruments FieldPoint Configuration Parameters Dialog Box 18-226
Figure 18-74. OPC Client Configuration Parameters Dialog Box 18-233
Figure 18-75. Neutralzone Definition Parameters Dialog Box 18-240
Figure 18-76. NIDAQ Device Configuration Parameters Dialog Box 18-242
Figure 18-77. NISCXI Definition Parameters Dialog Box 18-246
Figure 18-78. Omron Definition Parameters Dialog Box 18-251
Figure 18-79. OneShot Definition Parameters Dialog Box 18-255
Figure 18-80. OneShot Display Parameters Dialog Box 18-256
Figure 18-81. Optomux Definition Parameters Dialog Box 18-257
Figure 18-82. Numeric Only Pager Definition Parameters Dialog Box 18-263
Figure 18-83. Alphanumeric Pager Definition Parameters Dialog Box 18-263
Figure 18-84. Panel Definition and Display Parameters Dialog Box 18-267
Figure 18-85. Pareto Definition Parameters Dialog Box 18-276
Figure 18-86. Pareto Display Parameters Dialog Box 18-277
Figure 18-87. Unweighted Pareto Chart 18-278
Figure 18-88. Weighted Pareto Chart 18-278
Figure 18-89. Philips Configuration Parameters Dialog Box 18-282
Figure 18-90. PID Definition Parameters Dialog Box 18-286
Figure 18-91. Pipe Definition Parameters Dialog Box 18-292
Figure 18-92. Playwave Definitions Parameters Dialog Box 18-293
Figure 18-93. Pot Definitions Parameters Dialog Box 18-294
Figure 18-94. Pot Display Parameters Dialog Box 18-295
Figure 18-95. Profibus DP Configuration Parameters Dialog Box 18-298
Figure 18-96. ProfibusL2 Configuration Parameters Dialog Box 18-304
Figure 18-97. Pulse Definition Parameters Dialog Box 18-309
Figure 18-98. Pulse Display Parameters Dialog Box 18-310
Figure 18-99. Pushbutton Definitions Parameters Dialog Box 18-311
Figure 18-100. Verification Message Dialog Box 18-312
Figure 18-101. Pushbutton Display Parameters Dialog Box 18-313
Figure 18-102. Recipe Definition Parameters Dialog Box 18-325
Figure 18-103. Recipe Object Display Parameters Box 18-328
Figure 18-104. Reliance Definition Parameters Dialog Box 18-331
Figure 18-105. Run Definition Parameters Dialog Box 18-334
Figure 18-106. S5_3964 Definition Parameters Dialog Box 18-336
Figure 18-107. Sample Configuration Parameters Dialog Box 18-342
Figure 18-108. SampleText Definition Parameters Dialog Box. 18-344
Figure 18-109. Scale Definition Parameters Dialog Box 18-345
Figure 18-110. Scale Display Parameters Dialog Box 18-346
Figure 18-111. SiemensTI505 Definition Dialog Box 18-347

Figure 18-112. Sixnet Configuration Parameters Dialog Box 18-356
Figure 18-113. Spinner Definition Parameters Dialog Box 18-362
Figure 18-114. Spreadsheet Configuration Parameters Dialog Box 18-364
Figure 18-115. SqlExec Configuration Parameters Dialog Box 18-368
Figure 18-116. SquareD Definition Parameters Dialog Box Configured for Serial Communications 18-374
Figure 18-117. SquareD Definition Dialog Box Configured for SY/LINK Communications 18-376
Figure 18-118. SquareD Definition Dialog Box Configured for SY/ENET Communications 18-378
Figure 18-119. Switch Definition Parameters Dialog Box 18-381
Figure 18-120. Verification Message Dialog Box 18-381
Figure 18-121. Tesco Configuration Parameters Dialog Box 18-385
Figure 18-122. TextEntry Parameters Dialog Box 18-389
Figure 18-123. TextEntry Display Parameters Dialog Box 18-390
Figure 18-124. TimeOfDay Definition Parameters Dialog Box 18-392
Figure 18-125. TimeOfDay Display Parameters Dialog Box 18-393
Figure 18-126. Tiway Configuration Parameters Dialog Box 18-394
Figure 18-127. Toshiba M Series Configuration Parameters Dialog Box 18-403
Figure 18-128. Toshiba T Series Configuration Parameters Dialog Box 18-404
Figure 18-129. Trend definition parameters dialog box 18-408
Figure 18-130. Plot of a Logical Value 18-410
Figure 18-131. Trend Display Parameters Dialog Box 18-411
Figure 18-132. XBarR Definition Parameters Dialog Box 18-415
Figure 18-133. XBarR Display Parameters Dialog Box 18-416
Figure 18-134. X-Bar Chart Showing Upper Control Limit, Center Line, and Lower Control Limits. 18-417
Figure 18-135. R Chart Showing Upper Control Limit, Center Line, and Lower Control Limits as Calculated Based on Plotted Samples. 18-417
Figure 18-136. XChart Definition Parameters Dialog Box 18-421
Figure 18-137. XChart Display Parameters Dialog Box 18-422
Figure 18-138. XYChart Definition Parameters Dialog Box 18-424
Figure 18-139. XYChart Display Parameters Dialog Box 18-425

Tables

Table 2-1. Switch Object Class Database Members 2-5
Table 2-2. General Numeric Format 2-7
Table 2-3. Leading zeroes 2-7
Table 2-4. Fractional numbers with trailing zeroes 2-8
Table 2-5. Exponential/Scientific notation 2-8
Table 2-6. Hexadecimal formats 2-8
Table 2-7. Time Periods 2-9
Table 2-8. Absolute Dates and Times 2-10

Table 4-1. Spreadsheet Column Labels 4-15

Table 6-1. Dialing Prefix 6-6

Table 7-1. Arithmetic Operators 7-10
Table 7-2. Comparison Operators 7-11

Table 8-1. Tools for Displaying Dynamic Graphics 8-6

Table 9-1. Alarm Colors 9-6

Table 12-1. Special Access SQL Characters 12-3
Table 12-2. Data Transforms 12-4

Table 18-1. Allen-Bradley DH+ Interface Memory Addresses 18-6
Table 18-2. AB_PLC2 Data Members 18-10
Table 18-3. AB_SLC500 Data Members 18-10
Table 18-4. AB_PLC5 Data Members 18-13
Table 18-5. Accumulator Data Member 18-23
Table 18-6. Alarm Data Members 18-29
Table 18-7. $Alarm Data Members 18-31
Table 18-8. Alternator Data Members 18-34
Table 18-9. HOA Modes 18-37
Table 18-10. Animator Data Members 18-43
Table 18-11. Lookout Applicom Object Classes and the Corresponding Protocols/Devices 18-45
Table 18-12. Applicom Local Data Members 18-46
Table 18-13. Applicom JBUS Data Members 18-46
Table 18-14. Applicom April 1000 Data Members 18-47
Table 18-15. Applicom Klockner-Moeller Data Members 18-48
Table 18-16. Applicom Otic Fischer & Porter Data Members 18-49
Table 18-17. Applicom Profibus DP Data Members 18-50
Table 18-18. Applicom Profibus FMS Data Members 18-51
Table 18-19. Applicom Profibus L2 Data Members 18-52
Table 18-20. Applicom SAIA SBus Data Members 18-54
Table 18-21. Applicom Siemens S5 AS511 Data Members 18-55
Table 18-22. Applicom Siemens S7 MPI Data Members 18-56
Table 18-23. Applicom Siemens S7 PPI Data Members 18-58
Table 18-24. Applicom Siemens H1 Data Members 18-59
Table 18-25. Applicom Telemecanique Data Members 18-61
Table 18-26. Aquatrol Data Members 18-69
Table 18-27. ASCII Data Members 18-73
Table 18-28. Data Types Allowed by ASCII 18-77

Table 18-29. Average Data Members 18-82
Table 18-30. Counter Data Members 18-84
Table 18-31. Cutler-Hammer Data Member Set 18-87
Table 18-32. Column Names 18-100
Table 18-33. DataTable Data Members 18-100
Table 18-34. DdeLink Data Members 18-104
Table 18-35. DdeTable Data Members 18-108
Table 18-36. DelayOff Data Members 18-110
Table 18-37. DelayOn Data Members 18-112
Table 18-38. DeltaTau Data Members 18-114
Table 18-39. Derivative Data Members 18-116
Table 18-40. DialGauge Data Members 18-119
Table 18-41. DL205 and DL405 Data Members 18-123
Table 18-42. Dynamic Data Members 18-129
Table 18-43. ElapsedTime Data Members 18-133
Table 18-44. Event Data Members 18-134
Table 18-45. Expression Data Members 18-136
Table 18-46. FisherROC Data Members 18-139
Table 18-47. Flipflop Data Members 18-144
Table 18-48. Gauge Data Members 18-146
Table 18-49. GE_Series6 Data Members 18-149
Table 18-50. GE_Series90 Data Members 18-154
Table 18-51. Histogram Data Members 18-159
Table 18-52. Hitachi Data Members (Address Ranges in Hexadecimal) 18-163
Table 18-53. HyperTrend Data Members 18-171
Table 18-54. Integral Data Members 18-174
Table 18-55. Interpolate Data Members 18-178
Table 18-56. Interval Data Members 18-180
Table 18-57. IPASCII Data Members 18-182
Table 18-58. Data Types Allowed by IPASCII 18-185
Table 18-59. Junction Data Members 18-190
Table 18-60. $Keyboard Data Members 18-192
Table 18-61. LatchGate Data Members 18-194
Table 18-62. Maximum Data Members 18-195
Table 18-63. Minimum Data Members 18-197
Table 18-64. Mitsubishi Data Members (A Series) 18-201
Table 18-65. MitsubishiFX Data Members (FX Series) 18-201
Table 18-66. 6-Digit Address Coding 18-209
Table 18-67. Modbus Data Members 18-210
Table 18-68. ModbusMOSCAD Data Members 18-214
Table 18-69. ModbusSlave Data Members 18-217
Table 18-70. Multistate Data Members 18-220
Table 18-71. National Instruments Fieldbus Data Members 18-224

Table 18-72. FieldPoint Data Members 18-228
Table 18-73. Multiple Discrete Data Members 18-229
Table 18-74. OPC Client Data Members 18-235
Table 18-75. Neutralzone Data Members 18-241
Table 18-76. NIDAQDevice Data Members 18-243
Table 18-77. National Instruments Data Acquisition Devices Supported by Lookout 18-244
Table 18-78. NISCXI Device Data Members 18-247
Table 18-79. Omron Data Members 18-253
Table 18-80. OneShot Data Members 18-256
Table 18-81. Optomux Data Members 18-260
Table 18-82. Pager Data Members 18-265
Table 18-83. Panel Data Members 18-274
Table 18-84. Pareto Data Members 18-280
Table 18-85. Philips Data Members 18-284
Table 18-86. PID Data Members 18-290
Table 18-87. Pipe Data Members 18-292
Table 18-88. Playwave Data Members 18-293
Table 18-89. Pot Data Members 18-296
Table 18-90. Profibus DP Data Members 18-299
Table 18-91. ProfibusL2 Data Members 18-307
Table 18-92. Pulse Data Members 18-310
Table 18-93. Pushbutton Data Members 18-313
Table 18-94. RKC Data Member Group 18-316
Table 18-95. Measured Value Group 18-317
Table 18-96. Operation Mode Group 18-317
Table 18-97. Memory Area and Set Value Group 18-318
Table 18-98. Parameter Group 10 (Measured Input Parameters) 18-318
Table 18-99. Parameter Group 11(Remote Setting Input Parameters) 18-318
Table 18-100. Parameter Group 12 (Output Parameters) 18-319
Table 18-101. Parameter Group 13 (Auto-Tuning Bias Parameters) 18-319
Table 18-102. Parameter Group 14 (Alarm 1 Parameters) 18-319
Table 18-103. Parameter Group 15 (Analog Output Parameters) 18-320
Table 18-104. Parameter Group 16 (Positioning, Proportioning, PID Action Parameters) 18-320
Table 18-105. Parameter Group 17 (Bar Graph Parameter) 18-320
Table 18-106. Parameter Group 20 (Input Selection Parameters) 18-320
Table 18-107. Parameter Group 21 (Setting Parameters) 18-321
Table 18-108. Parameter Group 22 (Output Action Parameters) 18-321
Table 18-109. Parameter Group 23 (Alarm 2 Parameters) 18-321
Table 18-110. Parameter Group 40 (Data Lock Parameters) 18-322
Table 18-111. Lookout Data Members 18-322
Table 18-112. Recipe Data Members 18-329

Table 18-113. Reliance Data Members (all addresses are in octal).....18-333
Table 18-114. Run Data Members.....18-334
Table 18-115. S5_3964 Data Members 18-338
Table 18-116. Sample Data Members 18-342
Table 18-117. SampleText Data Members.....18-344
Table 18-118. Scale Data Members.....18-346
Table 18-119. SiemensTI505 Data Members 18-351
Table 18-120. Sixnet Data Members 18-358
Table 18-121. Spinner Data Members.....18-363
Table 18-122. Spreadsheet Data Members.....18-367
Table 18-123. SqlExec Data Members 18-369
Table 18-124. SquareD Data Members 18-379
Table 18-125. Switch Data Members 18-383
Table 18-126. $System Data Members.....18-384
Table 18-127. Tesco Data Members 18-387
Table 18-128. TextEntry Data Members.....18-391
Table 18-129. TimeOfxxxx Data Members.....18-393
Table 18-130. Tiway Data Members 18-398
Table 18-131. Toshiba M Series Data Members 18-406
Table 18-132. Toshiba T Series Data Members 18-406
Table 18-133. Trend Data Members.....18-412
Table 18-134. Wizdom Data Members 18-414
Table 18-135. XBarR Data Members 18-418
Table 18-136. Wizdom Data Members 18-422
Table 18-137. XYChart Data Members 18-425

About This Manual

Organization of the Product User Manual

The Lookout Reference Manual is organized as follows:

- Chapter 1, *Installing Lookout*, describes how to install your Lookout software.
- Chapter 2, *Introduction*, describes the Lookout architecture and presents the background necessary to fully understand how Lookout provides solutions for your continuous process, discreet, or batch applications.
- Chapter 3, *Getting Started*, explains how to start and get around within Lookout. It describes the Lookout screen and introduces some important mouse and keyboard shortcuts.
- Chapter 4, *Using Lookout*, discusses the basics of using Lookout, including selecting objects, creating objects, editing object databases, and other operations.
- Chapter 5, *Developer Tour*, guides you through the development of a simple process file.
- Chapter 6, *Serial Communications*, describes serial communications and details how to define settings for three different serial connections: hardwired, radio (RTS/CTS), and dial-up.
- Chapter 7, *Expressions*, explains the features and uses of Lookout expressions, which are flexible, real-time math statements.
- Chapter 8, *Graphics*, describes adding static and dynamic graphics to a control panel and creating and using custom graphics.
- Chapter 9, *Alarms*, describes generated alarms and configuration services provided by the Lookout alarm subsystem. As a Lookout environment service, the alarm subsystem filters, displays, logs, and prints alarms.
- Chapter 10, *Security*, describes Lookout accounts and the environment service security subsystem, which oversees process file security, control security, viewing security, and action verification. With this system, you selectively determine which operators control particular objects, which operators view particular control panels, and which objects prompt the operator for command verification.

- Chapter 11, *Logging Data and Events*, describes three Lookout methods for logging real-time system data to disk—Spreadsheet Logger, Citadel Threaded Database Logger, and Event Logger—and report generation.
- Chapter 12, *Structured Query Language*, describes Structured Query Language (SQL), Open Database Connectivity (ODBC), and accessing Citadel data using both SQL and ODBC.
- Chapter 13, *Dynamic Data Exchange*, explains how to use Dynamic Data Exchange (DDE) with Lookout. DDE is the Microsoft message-based protocol used by applications like Microsoft Excel and Lookout to link to data in other applications.
- Chapter 14, *Networking*, explains how to use Lookout to monitor and control your process from any workstation (node) on the network.
- Chapter 15, *Redundancy*, describes how to configure two process control computers for redundancy, providing automatic transfer of control should the primary computer fail.
- Chapter 16, *Runtime Menu Commands*, describes Lookout menu bar pull-down commands available in Normal mode (that is, not Edit) mode.
- Chapter 17, *Edit Mode Menu Commands*, describes Lookout menu bar pull-down commands available in Edit mode. You use Edit mode to perform all system configuration and editing.
- Chapter 18, *Object Class Reference*, describes Lookout object classes, listed in alphabetical order. Input parameter syntax and data members are documented for each object class, along with a description of the functionality of each object class and an example of how to use the object.
- Appendix A, *Customer Communication*, contains forms you can use to request help from National Instruments or to comment on our products and manuals.
- The *Index* contains an alphabetical list of key terms and topics in this manual, including the page where you can find each one.

Conventions Used in This Manual

The following conventions are used in this manual:

<>	Angle brackets enclose the name of a key on the keyboard—for example, <Shift>. Angle brackets containing numbers separated by an ellipsis represent a range of values associated with a bit or signal name—for example, DBIO<3..0>. Key names are capitalized
-	A hyphen between two or more key names enclosed in angle brackets denotes that you should simultaneously press the named keys—for example, <Control-Alt-Delete>.
»	The **»** symbol leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options» Substitute Fonts** directs you to pull down the **File** menu, select the **Page Setup** item, select **Options**, and finally select the **Substitute Fonts** options from the last dialog box.
☞	This icon to the left of bold italicized text denotes a note, which alerts you to important information.
⚠	This icon to the left of bold italicized text denotes a caution, which advises you of precautions to take to avoid injury, data loss, or a system crash.
bold	Bold text denotes the names of menus, menu items, parameters, dialog boxes, dialog box buttons or options or data entry fields, icons, windows, or Windows 95 tabs.
bold italic	Bold italic text denotes a, note, caution, warning, comment, or related objects or functions.
italic	Italic text denotes variables, emphasis, a cross reference, or an introduction to a key concept. This font also denotes text from which you supply the appropriate word or value, as in Windows 3.*x*.
`monospace`	Text in this font denotes text or characters you would type, such as the contents of a dialog box data entry field, or expression, Lookout data members, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, functions, operations, variables, and filenames and extensions.
`monospace italic`	Italic text in this font denotes that you must enter the appropriate words or values in the place of these items.

Related Documentation

Lookout Object Developer's Toolkit Reference, National Instruments Part Number 321709A-01

Customer Communication

National Instruments wants to receive your comments on our products and manuals. We are interested in the applications you develop with our products, and we want to help if you have problems with them. To make it easy for you to contact us, this manual contains comment and configuration forms for you to complete. These forms are in Appendix A, *Customer Communication*, at the end of this manual.

Part I

Getting Started

Chapters 1 through 5 contain information to help you get started quickly and easily with Lookout.

In particular, Chapter 2, *Introduction*, and Chapter 5, *Developer Tour*, contain both theory and practical examples to help you learn how to quickly become productive using Lookout. Chapter 3, *Getting Started*, and Chapter 4, *Using Lookout*, are helpful in familiarizing you with the Lookout interface and learning how to use the Lookout tools.

- Chapter 1, *Installing Lookout*, describes how to install your Lookout software.
- Chapter 2, *Introduction*, describes the Lookout architecture and presents the background necessary to fully understand how Lookout provides solutions for your continuous process, discreet, or batch applications.
- Chapter 3, *Getting Started*, explains how to start and get around within Lookout. It describes the Lookout screen and introduces some important mouse and keyboard shortcuts.
- Chapter 4, *Using Lookout*, discusses the basics of using Lookout, including selecting objects, creating objects, editing object databases, and other operations.
- Chapter 5, *Developer Tour*, guides you through the development of a simple process file.

1

Installing Lookout

Hardware Requirements

Lookout requires a 386-, 486-, or Pentium class IBM PC-compatible personal computer. The computer should have at least 8 MB of RAM memory and 400 MB of free hard disk space. It should have a CD-ROM drive, a floppy drive, a mouse (preferably bus version), and video display/controller combination with at least VGA (640 × 480) capability.

Because Lookout can run 24 hours a day, your computer should have some form of AC power surge protection. An uninterruptible power supply (UPS) provides the ultimate protection. A UPS provides complete isolation between the AC power source and the computer and has backup battery power if there are blackouts and brownouts. A quality surge protector will protect your computer from most electrical surges and spikes if you do not need battery backup.

Software Requirements

Lookout 3.8 includes both 16-bit and 32-bit versions. Future releases of Lookout will include 32-bit versions only.

16-bit Lookout software can run on Microsoft Windows 3.1 or higher, Windows for Workgroups 3.11 or higher, and Windows 95. It requires DOS 3.0 or higher.

While you can run 16-bit Lookout on a Windows NT system, National Instruments recommends that you use the 32-bit version.

32-bit Lookout software can run on Windows 95 and Windows NT.

Installing Lookout

To Install Lookout

1. Insert Lookout Disk 1 into drive A or drive B on your computer. If you are installing from a CD, insert the CD into your CD drive.
2. If you are using Windows 95, click the Start button, then click on **Run…**.

   If you are using Windows 3.*x* or Windows NT, select **File»Run…** from the Windows Program Manager.

3. Enter A:\SETUP if you are using drive A. Enter B:\SETUP if you are using drive B, and so on. Then select **OK**.

   If you are installing Lookout on a 16-bit Windows platform such as Windows 3.1 or 3.11, Setup automatically installs 16-bit Lookout.

   If you are installing Lookout on a 32-bit platform such as Windows 95 or Windows NT, Setup gives you the option of installing either the 16-bit or 32-bit version of Lookout. While the 16-bit version of Lookout runs on 32-bit platforms, it is best to load the 32-bit version if you are running windows 95 or Windows NT.

4. If prompted, choose either 16-bit or 32-bit Lookout.

   Next, Lookout gives you the option of installing its ODBC driver. (This driver is required if you wish to query the Citadel historical database using SQL). See Chapter 12, *Structured Query Language*, for more information regarding this feature.

5. Shut down all applications that may currently be using ODBC. Such applications include spreadsheets, word processors, database programs, MS Query, and similar applications.

6. Choose 16-bit ODBC, 32-bit ODBC, or none—you can install it later if you like.

   If you are installing Lookout on a 16-bit Windows platform such as Windows 3.1 or 3.11, Setup automatically installs the 16-bit ODBC driver.

   As with Lookout itself, if you are installing on a 32-bit platform such as Windows 95 or Windows NT, you can install either the 16-bit or 32-bit version of the ODBC driver. You should load the 16-bit driver if you loaded the 16-bit version of Lookout, and the 32-bit driver if you loaded the 32-bit version of Lookout.

   Setup prompts you for a hard disk subdirectory name in which to install the Lookout files.
7. At this point in the installation, you can choose whether or not to install the Lookout online help.
8. Use **Browse** to enter the name of an alternate directory or select **OK** to accept the recommended directory name.
9. Follow the remaining instructions to complete the Lookout installation.

Starting Lookout for the First Time

The first time you launch Lookout, it prompts you for registration information.

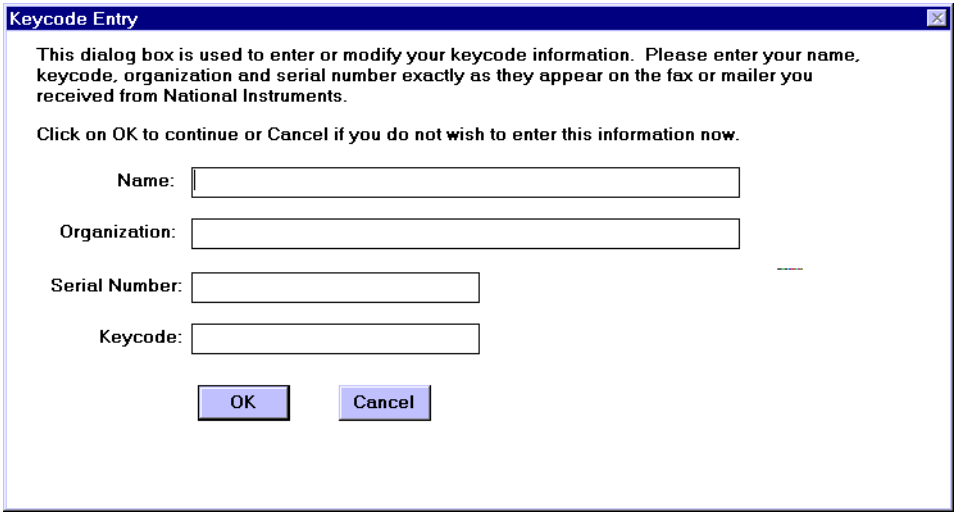
☞ **Notes** ***Be sure to register your Lookout Package to receive your permanent unlock code! As an unregistered package, Lookout is limited to 50 I/O points and only runs for 30 days. When you register Lookout, you unlock it for permanent use at your appropriate I/O count. If you do not register Lookout by the end of the 30 day period, it lapses to a demo system. You must complete the license agreement and mail or fax a copy of the agreement to National Instruments in order to register Lookout. Upon receipt of the registration form, National Instruments generates a key code to unlock Lookout and faxes or mails it to you.***

Lookout requires a hardware key in some countries. Contact National Instruments if you are not sure whether your system requires a hardware key. If you were supplied a key with Lookout, be sure to plug it into the parallel port on your computer before activating Lookout.

To Start Lookout for the First Time

1. Launch Lookout by selecting **Start»Programs»Lookout»Lookout**, or double-click on the Lookout icon.

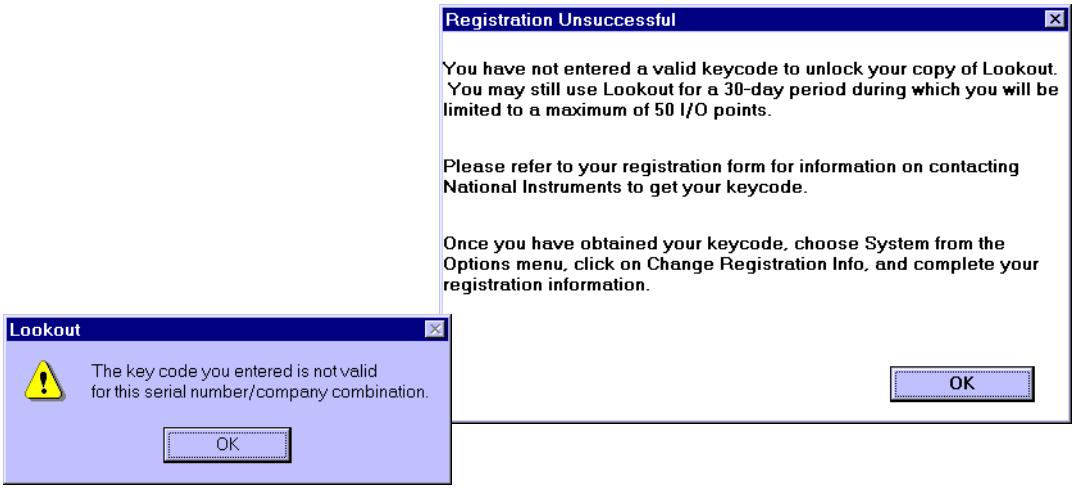
   The registration dialog box appears.


Keycode Entry
This dialog box is used to enter or modify your keycode information. Please enter your name, keycode, organization and serial number exactly as they appear on the fax or mailer you received from National Instruments.
Click on OK to continue or Cancel if you do not wish to enter this information now.
Name:
Organization:
Serial Number:
Keycode:
OK
Cancel

2. Enter your name in the **Name** field.
3. Enter the **Organization** name exactly as it appears on the key code fax sent in response to your registration, including punctuation marks. This text is used in combination with the key code and so it must be exact.
4. Enter the **Serial Number** of your package. (This can be found on your registration form.)
5. Enter your 12-character **Keycode**. The key code is not case sensitive and you can leave the hyphens out if desired. Notice that there are no spaces near the hyphens.
6. After completing the entries, press <Enter> or select **OK**.

   If you enter the proper information correctly, Lookout appears on your screen with no process running.

   If you have not yet received your key code from National Instruments, you can enter your registration data later. Select **OK** and Lookout will inform you that you have not registered your package yet. Select **OK** again until Lookout launches.

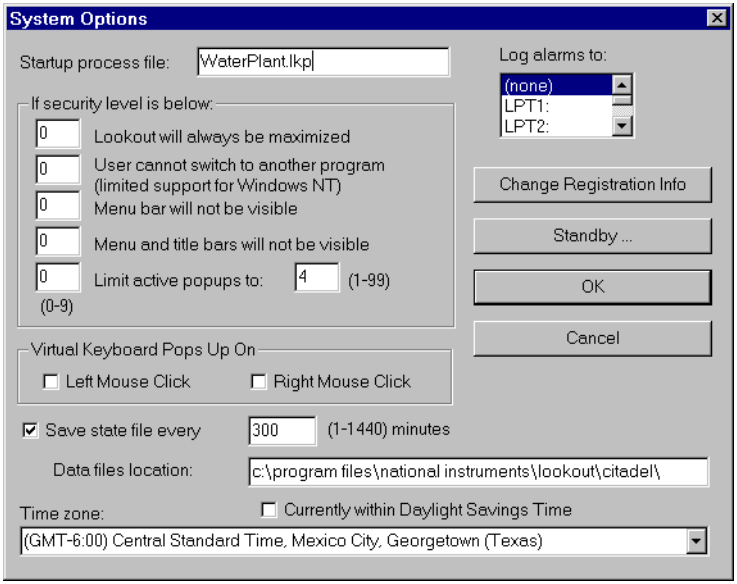

Registration Unsuccessful
You have not entered a valid keycode to unlock your copy of Lookout. You may still use Lookout for a 30-day period during which you will be limited to a maximum of 50 I/O points.
Please refer to your registration form for information on contacting National Instruments to get your keycode.
Once you have obtained your keycode, choose System from the Options menu, click on Change Registration Info, and complete your registration information.
OK
Lookout
The key code you entered is not valid for this serial number/company combination.
OK

If you are certain that you typed the information correctly and Lookout still does not accept it, call the National Instruments technical support line for help.

Automatic Process Loading

If your computer runs Lookout 24 hours a day, you may want to ensure that, if the computer temporarily loses power, it will automatically reboot and begin executing the process when power returns.

Enter the name of the process file you want to load automatically when Lookout begins in the **Startup process file** field of the **Systems Option** dialog box.



To make sure Lookout loads and runs when your computer boots or reboots, consult your operating system documentation instructions on how to set a default startup application.

2

Introduction

Lookout is a powerful yet easy-to-use MMI and SCADA software package for industrial automation. Lookout runs under Windows and communicates with field I/O from PLCs, RTUs, and other devices. Typical Lookout projects include continuous process monitoring and supervisory control, discrete manufacturing, batch applications, and remote telemetry systems.

Thoroughly object-oriented and event driven, Lookout is a configurable package that requires no programming or scripting—just fill in the blanks.

With Lookout, you can create graphical representations on a computer screen of real-world devices such as switches, dial gauges, chart recorders, pushbuttons, knobs, sliders, meters, and then link your images to the actual field instruments using PLCs, RTUs, DAQ boards, or other I/O devices. You can configure Lookout to generate alarms, log data to disk, animate custom graphics, print reports, automatically adjust setpoints, historically trend information, and warn operators of malfunctions.

Lookout has many diverse capabilities such as Statistical Process Control (SPC), recipe management, Structured Query Language (SQL), built-in security, flexible data logging, sophisticated animation, complex alarming, radio and dial-up telemetry support, audit trails of events and setpoint adjustments, multimedia support, touch screen compatibility, DDE & NetDDE, and more.

With Lookout you can develop an application completely on-line, without shutting down. You do not have to recompile or download a database every time you make a modification, nor do you have to switch back and forth between programs. You do not even have to run separate development and configuration programs. Instead, you can add, delete and modify control panels, logic, graphics, PLCs, RTUs, I/O, and other field devices without ever interrupting your process.

Because Lookout is object-oriented and event-driven, you can use Lookout with other programs in the Microsoft Windows multitasking environment. for example, while Lookout monitors and controls your process, you can use a spreadsheet to analyze production figures of hourly average flow rates, then start a word processor to generate a memorandum, paste the spreadsheet into the memo and send it to a laser printer.

The remainder of this chapter describes the Lookout architecture and presents the background necessary to fully understand how Lookout provides solutions for your continuous process, discreet, or batch applications.

Architecture

Once you understand the basic Lookout basic components, and the fundamentals regarding object-oriented and event-driven structure, using the program becomes much easier.

What is an Object?

A Lookout object is a self-contained software unit that has a predefined database, a set of parameters, and built-in functionality. The following diagram depicts the functionality, data members and parameters of an object.

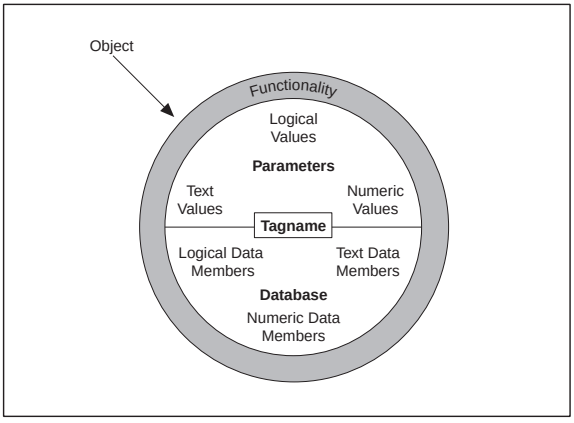


Figure 2-1. An object encapsulates data, parameters, and functionality in one bundle.

Think of an object as a software model of something physical. For example, a light switch is something physical. You can turn it on and off. In Lookout, a Switch object represents the physical switch. You can turn it on and off, too.

Parameters define the limits of object functionality. For example, the Switch object **Security Level** parameter determines who can turn it on and off. The object *database* stores information indicating the current switch position.

Functionality

Different object classes are designed to perform different functions. For example, the Pot (potentiometer) object class operates differently from the Switch object class. This is the *functionality* built into every object class.

Functionality is the way an object works, operates, or performs a task. Functionality is a general concept that applies in the same way to all objects in a given object class. Parameters, however, can be unique, and define the specific functionality of an individual object.

The object class definitions featured in Chapter 18, *Object Class Reference*, outline the functionality of a given object class.

Parameters

The object parameters define its characteristics. Lookout uses object parameters to complete the definition of the object functionality. For example, **Data rate**, **Parity**, and **Stop bits** are a few of the parameters that define how a Modbus object works. Other examples include the **Control security level** of a Switch object; **Minimum**, **Maximum**, and **Resolution** of a Pot object; and **Data** of an Average object.

Every object class supports a set of parameters that you must fill in or select when creating a new object. Some parameters support *expressions*, which can be variable. Others require constant values. Some ask you to pick specific settings.

Parameters that accept expressions appear as yellow data entry fields. These parameters can receive signals (that is, they are writable). See Chapter 7, *Expressions*, for more detailed information on expressions.

All the parameters for any given class are visible in the object definition dialog box.

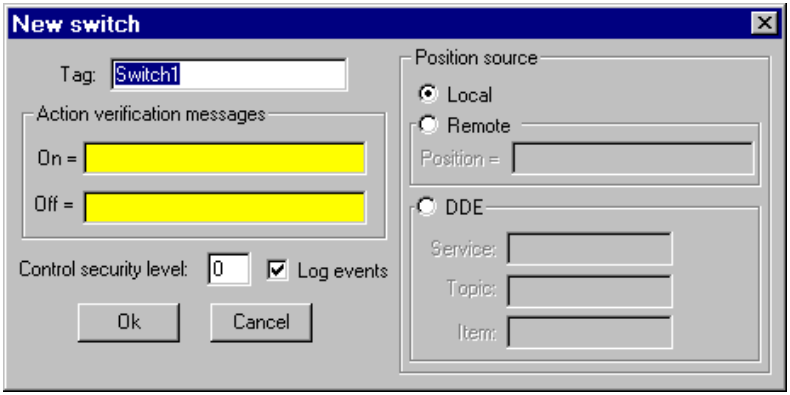


Figure 2-2. Switch Definition Dialog Box

Database

Each object has its own built-in database. The individual parts of this self-contained database are called *data members*. Some object classes (that is, types of objects) have a very limited database, while others have extensive databases. The database of an object representing a PLC might have hundreds of data members; but a Switch object database has only three data members. You do not have to build a database—the data members are automatically available when you create the object.

In the case of a switch, the implicit value of the object is a part of the self-contained database. Data members can either generate (write) signals, receive (read) signals, or both.

Every data member contains a single value that can be one of three types: numeric, logical, or text. They must read or write compatible signals. For example, you cannot connect a data member that generates a logical signal to a data member that receives a numeric signal.

Lookout performs strong type checking when you connect objects. It will generate an error message if you try to connect incompatible signals.

The built-in data members for each object are referred to as *native* members, and can be thought of as the default or automatic data members. You can add to and modify the database to suit your specific needs.

Every object class has an explanation of its database located in its definition. The following excerpt is an example of the switch object class database explanation.

Table 2-1. Switch Object Class Database Members

Data Members	Type	Read	Write	Description
(implicit)	logical	yes	no	switch position
enable	logical	no	yes	If set to TRUE, enables DDE. If set to FALSE, disables DDE. The default value is on. This data member does not interfere with existing code.
visible	logical	no	yes	When false, the switch object cannot be seen on the display panel. When true, the switch can be seen and controlled.

Data Members

Data members are the individual parts of the self-contained object database. There are three types of data members: Logical, Numeric, Text. Some object classes also have (implicit) data members.

Logical Data Members

Logical data members contain a value that represents a binary or on/off state. A light switch is a logical device—it is either on or off.

Logical data members are typically used to control equipment that can be turned on and off, to indicate that a piece of equipment is running, or indicate whether a limit switch is open or closed.

The Switch object generates a logical signal that is on when the switch is up and off when the switch is down. In the same way, the Pushbutton object generates a logical value that is on while the pushbutton is depressed. The Pulse object generates a logical signal that turns on and off at prescribed intervals, creating a logical pulse.

The logical signals that some objects generate can be displayed graphically on a control panel. See Chapter 8, *Graphics*, for more information.

Lookout recognizes the following logical constants as expressions:

Logical constants that represent an on state: `yes`, `true`, `on`

Logical constants that represent an off state: `no`, `false`, `off`

Numeric Data Members

A *numeric data member* is a floating point number representing analog values such as tank level, pressure, flow rate, voltage, and temperature. Numeric data members also represent time, either as a time period (span) or as an absolute time (that is, a particular time of day/week/month/year).

The Pot (potentiometer) object generates a numeric signal compatible with the numeric signals that monitor and control the analog input and output points on a PLC. Numeric signals range from -1.7×10^{308} to 1.7×10^{308}, and the magnitude can be as small as 1.7×10^{-308}.

Numeric constants are entered using decimal digits (0 – 9), the minus sign (–), the exponent symbol (E or e), and the time format separator (:).

Examples of numeric constants
0
–123.779999
1.5E7 = 15,000,000
–3.7E–3 = –0.0037
–.0036
123356636.2345791

Time or *Time signals* are stored by Lookout as numeric values that represent days and fractions of a day. For example, you enter one hour as 1:00:00. Lookout interprets the number to the right of the rightmost colon (:) as seconds, the number to the right of the second colon from the right as minutes, the next number as hours, and the number to the left of the third colon from the right as days. If there are no colons in the entry, the time period is assumed to be given in days.

Examples of numeric time period constants	
0:23	= 23 seconds, or 0.0002662 days
75:00	= 75 minutes, 0 seconds, or 0.05208 days
12:00:05:01	= 12 days, 0 hours, 5 minutes, 1 second, or 12.003484 days

Examples of numeric time period constants	
199::	= 199 hours, 0 minutes, 0 seconds, or 8.2917 days
0:10.023	= 10.023 seconds, or 0.0001160 days
12.75	= 12.75 days
17:64:22.5	= invalid number, because hours are specified: minutes must be <= 59

You may enter one hour as 1:00:00, but Lookout stores the number as 0.04167 (or 1/24 of a day). Days are represented by the integer portion of the number. The number zero represents Jan. 1, 1900.

You can display the numeric signals that some objects generate on a control panel. See Chapter 8, *Graphics*, for more information.

If you display the numeric signal digitally, Lookout provides a long list of numeric formats to choose from. These are listed below.

Table 2-2. General Numeric Format

(General)	number displayed in most compact form possible 123.789 displayed as 123.789

Table 2-3. Leading zeroes

000000000	123.789 displayed as 000000124
00000000	123.789 displayed as 00000124
0000000	123.789 displayed as 0000124
000000	123.789 displayed as 000124
00000	123.789 displayed as 00124
0000	123.789 displayed as 0124
000	123.789 displayed as 124
00	123.789 displayed as 124
0	123.789 displayed as 124

Table 2-4. Fractional numbers with trailing zeroes

0.0	123.789 displayed as 123.8
0.00	123.789 displayed as 123.79
0.000	123.789 displayed as 123.789
0.0000	123.789 displayed as 123.7890
0.00000	123.789 displayed as 123.78900
0.000000	123.789 displayed as 123.789000
0.0000000	123.789 displayed as 123.7890000
0.00000000	123.789 displayed as 123.78900000

Table 2-5. Exponential/Scientific notation

0E0	123.789 displayed as 1E+2
0.0E+0	123.789 displayed as 1.2E+2
0.00E+0	123.789 displayed as 1.24E+2
0.000E+0	123.789 displayed as 1.238E+2
0.0000E+0	123.789 displayed as 1.2379E+2
0.00000E+0	123.789 displayed as 1.23789E+2
0.000000E+0	123.789 displayed as 1.237890E+2
0.0000000E+0	123.789 displayed as 1.2378900E+2
0.00000000E+0	123.789 displayed as 1.23789000E+2

Table 2-6. Hexadecimal formats

0x0	123.789 displayed as 0x7B
0x00	123.789 displayed as 0x7B
0x000	123.789 displayed as 0x07B
0x0000	123.789 displayed as 0x007B
0x00000	123.789 displayed as 0x0007B
0x000000	123.789 displayed as 0x00007B

Table 2-6. Hexadecimal formats (Continued)

0x0000000	123.789 displayed as 0x000007B
0x00000000	123.789 displayed as 0x0000007B

You can also use numeric signals to represent absolute times and periods of time. Because dates and times are represented by numeric values, you can add, subtract, and include dates and times in expressions, just as you would any other numeric signals.

A time *period* represents a span of time or a duration. Time periods are indicated in hours, minutes, seconds, and fractions of seconds. Numeric formats that represent time periods are characterized by capital letters (that is, H rather than h).

Table 2-7. Time Periods

H	0.4789 displayed as 11 (hours)
H.H	0.4789 displayed as 11.5 (hours)
H.HH	0.4789 displayed as 11.49 (hours)
M	0.4789 displayed as 690 (minutes)
M.M	0.4789 displayed as 689.6 (minutes)
M.MM	0.4789 displayed as 689.62 (minutes)
S	0.4789 displayed as 41377 (seconds)
S.S	0.4789 displayed as 41377.0 (seconds)
S.SS	0.4789 displayed as 41376.96 (seconds)
HH:MM	0.4789 displayed as 11:29 (11 hours, 29 minutes)
HH:MM:SS	0.4789 displayed as 11:29:36 (11 hours, 29 min, 36 seconds)
HH:MM:SS.S	0.4789 displayed as 11:29:36.9
HH:MM:SS.SS	0.4789 displayed as 11:29:36.96
HH:MM:SS.SSS	0.4789 displayed as 11:29:36.960
MM:SS	0.4789 displayed as 689:36 (689 minutes, 36 seconds)
MM:SS.S	0.4789 displayed as 689:36.9

Table 2-7. Time Periods (Continued)

MM:SS.SS	0.4789 displayed as 689:36.96
MM:SS.SSS	0.4789 displayed as 689:36.960

Absolute dates and times indicate a specific moment in time. Lookout stores all absolute dates and times as numeric signals. It uses the 1900 date system in which the number 1 corresponds to midnight, January 1, 1900. The number 2 corresponds to midnight, January 2, 1900 and so on. For example, the number 34491.5 represents noon, June 6, 1994.

Numeric formats that represent absolute times are characterized by lower case letters (for example, hh:mm instead of HH:MM).

Table 2-8. Absolute Dates and Times

hh:mm	34668.7889 displayed as 18:56 (6:56 p.m.)
hh:mm:ss	34668.7889 displayed as 18:56:02
mm/dd hh:mm	34668.7889 displayed as 11/30 18:56
mm/dd hh:mm:ss	34668.7889 displayed as 11/30 18:56:02
mm/dd/yy	34668.7889 displayed as 11/30/94
mm/dd/yy hh:mm	34668.7889 displayed as 11/30/94 18:56
mm/dd/yy hh:mm:ss	34668.7889 displayed as 11/30/94 18:56:02
dd/mm hh:mm	34668.7889 displayed as 30/11 18:56
dd/mm hh:mm:ss	34668.7889 displayed as 30/11 18:56:02
dd/mm/yy	34668.7889 displayed as 30/11/94
dd/mm/yy hh:mm	34668.7889 displayed as 30/11/94 18:56
dd/mm/yy hh:mm:ss	34668.7889 displayed as 30/11/94 18:56:02

Text Data Members

Text data members contain text character strings. These character strings consist of all displayable characters. You can use text signals to display alarm descriptions on the alarm panel, to display labels on a control panel, and in parameters or expressions. You can enter text signals as constants, or you can construct them with the many text functions available in expressions. Be sure to enclose text constants within quotes (“ ”) when using them within expressions.

Examples of Text Constants
“Water Temperature:”
“ ” (empty text string)
“Low level in ‘Polymer 2’ tank”
“gpm”

(implicit) Data Members

Many object classes have an (implicit) data member. This implicit value is either logical, numeric, or textual, depending on the object class, and follows the same rules that apply to all other data members. The implicit member represents what Lookout considers to be the most commonly used data member of that object class. In many cases, it is the only data member of a class. It saves you time, and reduces the amount of typing required to designate a data member.

For example, Lookout could make you specify the numeric signal generated by a pot object by typing `Pot1.numeric` where `Pot1` is the tagname and `numeric` is the current value of the pot. Instead, you enter `Pot1` and Lookout knows you are referring to the implicit value of the pot. If you examine the Pot definition in Chapter 18, Object Definitions, you will see that the (`implicit`) data member is the current value of the pot.

Object Classes

An object is an individual instance of a particular object class. For example, Lookout has both Pot and Switch object classes, from which you might create 20 pots and 30 switches. In this case, you would be creating a total of 50 objects using only two object classes.

Lookout has an extensive library of object classes. To create an object, select the desired object class from the list that appears when you choose Create. This defines the type of object you want. Then you give the object a unique name (a tagname) and define its parameters.

Global object classes are a special kind of object class. Each contains global system data such as the number of currently active alarms. You cannot create, modify or delete a global object, but you can use its data members just as you would use any other object data members.

When you create or open a Lookout process file, Lookout automatically creates three global objects: $Alarm, $Keyboard, and $System.

Object Connections

In Lookout, you connect objects, allowing signals to pass between the objects—much the same way you would wire a time delay relay to a motor starter relay, for instance. You can do this by connecting database members together, or by connecting database members to parameters.

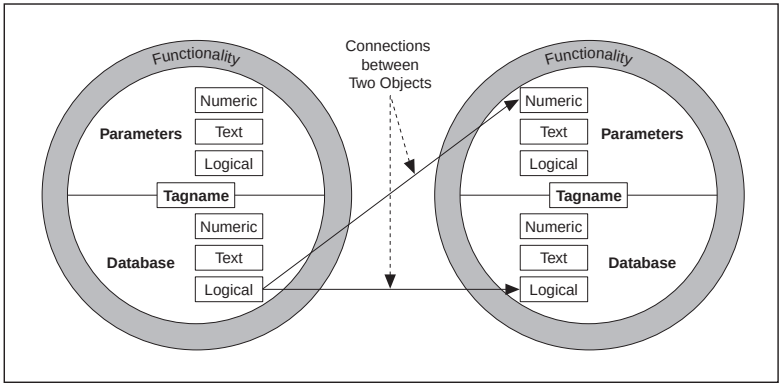


Figure 2-3. Example connections between two objects

For example, you might make the numeric data member of a Pot object the source for the **High Limit** parameter of an Alarm object. When you adjust the pot, the Alarm **High Limit** changes.

Supervisory Control

As you create and connect objects, you form a web, or system, containing many objects, all linked to perform a supervisory control strategy.

It is how you design your supervisory system, behind the control panel, that makes your process run. Your system routes signals from field components to bar graphs and visual indicators on control panels. It activates and deactivates alarms. You can design it to make complex decisions based on the values of field control signals and setpoints adjusted through pots and switches on control panels. You can include complex spreadsheet-style formulas as a part of your supervisory design. See Chapter 5, *Developer Tour*, for more detailed information on how to create objects and connect them together.

Event-Driven Processing

An important concept to understand is that Lookout is entirely *event-driven*, not *loop-driven*. To understand the significance of this design requires a digression, to explain how standard loop-driven programs work.

Loop-driven applications execute code sequentially from top to bottom, and loop back to the top to execute the same code over and over.

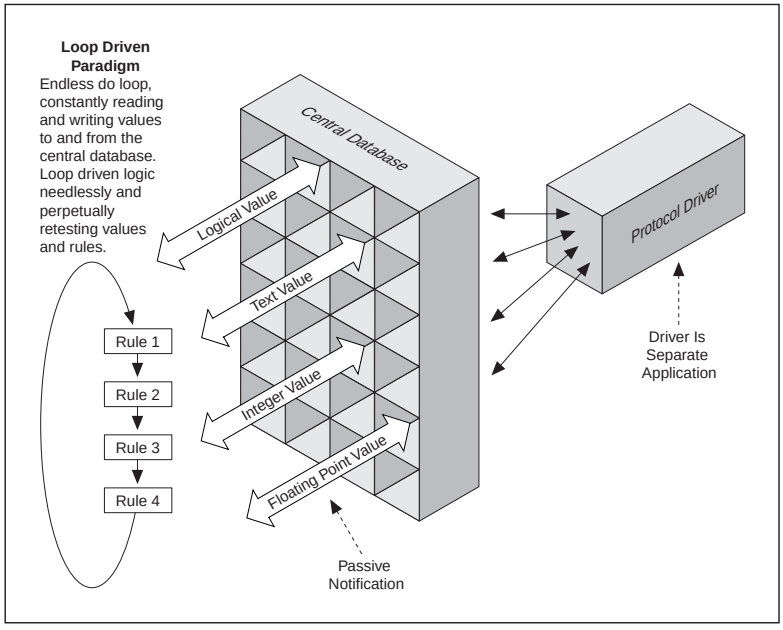


Figure 2-4. Example of conventional, Loop-Driven software program

In this programming model, a given section of code or rule may execute millions of times before the result changes. This wastes computer processor time and slows down responses to frequent events.

The more rules you add to loop-based systems, the slower the response time. Also, as you add more tagnames to the central database, speed and overall performance degrades. This is because many central databases use a *passive notification* system in which the rules of the loop-driven logic must scan an ever larger database for their appropriate values. The larger the database, the longer it takes the rule to find the data it needs to resolve its function.

In sharp contrast to this programming model, Lookout is entirely *event-driven*. Each object remains quiescent, doing nothing until an event occurs.

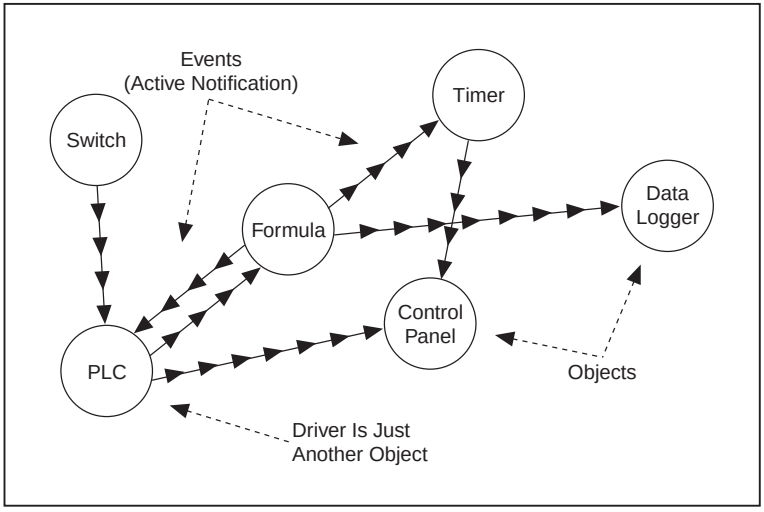


Figure 2-5. Lookout object-oriented and event-driven architecture

An *event* is a change in a data value. When an incoming signal changes, the object activates, processing the value according to its functionality. Objects only send out signals when the result of their processing changes. This is how an event propagates throughout your system, creating a chain-reaction that affects only the objects in the chain. This is called *active notification*.

Individual objects activate only when notified of an event. This event-driven paradigm closely simulates the responsiveness of physical switches, pushbuttons, and relays, and is the reason Lookout is so fast.

Advantages of Active Notification

Consider the advantages of active notification over passive notification. Active notification is inherently event-driven while passive notification requires a constant do-loop to check for a change in a data value.

A good example would be two people trying to exchange information. There are only two ways for the first to send data to the second.

One way is for person two to ask person one for information, and for person one to respond. The requires one person to ask the other repeatedly if

anything has changed (in other words, has an event occurred?). This is what loop-driven rules do when they constantly query the central database.

You can imagine the wasted time and effort if every once every minute, someone asked you to tell them about the latest events of the day—especially if your response was almost always “nothing new to report.”

On the other hand, person one could just tell person two when something happens (that is, when an event occurs). This is what happens when you connect two objects in Lookout.

As you can imagine, adding objects in Lookout adds significantly less demand for processor time than adding new rules or enlarging the central database of a traditional loop-driven system.

Because Lookout is entirely event-driven, the order in which you create the objects and connect them does not affect how your supervisory strategy works.

Environment Services

While the cornerstone of object technology is the object itself, objects need an environment in which to function. Objects require the use of system resources like the serial port, hard disk, multimedia functions, and more. For example, multiple PLC objects may need to use the same communication port on your computer. In such a situation, Lookout must provide a *service*—that is, a mechanism the objects can use to gain access to the communication port in an orderly and timely fashion.

A service, then, is a tool that Lookout makes available to objects, or a function Lookout performs outside of its object-oriented structure. Each service provides a special function. The following sections describe Lookout environment services.

Serial Port Communication Service

You can configure certain protocol object classes to represent and communicate with PLCs and RTUs through the serial ports of your computer. This service arbitrates serial port usage between objects representing PLCs and RTUs. For example, a single two-way radio connected to a serial port on the computer can communicate with several different brands of RTUs out in the field, each one using a different protocol. See Chapter 6, *Serial Communications*, for more information on configuring communications.

Database Service

With the database service you can define or modify native data member parameters.

For example, the Modbus object class includes a native data member called 40001. You can give this native data member an alias (or nickname) such as `PumpSpeed`, and define associated unit conversions, alarming, deadband, and other parameters.

With Lookout you can also directly import database information from external packages like Siemens APT. See Chapter 5, *Developer Tour*, for more information on Database services provided by Lookout.

Graphics Service

Lookout has an extensive library of standard graphics. These include various switches, potentiometers, pushbuttons, bar graphs, valves, tanks, pumps, and so on. You can also create your own custom graphic and add it to your Lookout library. See Chapter 8, *Graphics*, for more information on graphics.

Alarm Service

The alarm subsystem is a powerful and flexible mechanism for generating, displaying, logging, and printing alarms. This subsystem has several distinct parts including the alarm window, object parameters like **Alarm Group** and **Alarm Priority**, alarm filters, display parameters, and print settings.

Lookout permanently archives alarms to disk. You can easily print this alarm history. See Chapter 9, *Alarms*, for additional information.

Multimedia Service

Lookout also provides a multimedia service you can use to play sound wave files.

Security Service

Lookout has a highly sophisticated and comprehensive three-tier security system. The tiers include control security, viewing security, and action verification. You can selectively determine which operators have control of what objects, which operators can view what control panels, and which objects prompt operators for verification of commands. See Chapter 10, *Security*, for more information on Security.

Historical Logging Service

With the Logging service you can store real-time system information to disk in comma-delineated ASCII files, or in a special Lookout database called Citadel.

The Lookout Event Logger keeps track of who did what, and when they did it. Lookout logs operator commands, from closing a process file to flipping a switch or adjusting a pot. Along with each event, Lookout logs the account name (operator), date and time of the event, tagname of the object adjusted, and the before and after settings of the object. See Chapter 11, *Logging Data and Events*, for additional information on Logging services.

ODBC Service

Because of the Lookout Open Database Connectivity (ODBC) service, you can use other applications, such as Microsoft Access, to query the Lookout historical database. See Chapter 12, *Structured Query Language*, for additional information on ODBC services.

DDE Service

Lookout can send its live process values to other applications, and it can receive real-time values from other applications. The Lookout system acts as both a DDE client and a DDE server. See Chapter 13, *Dynamic Data Exchange*, for further information on DDE.

Networking Service

Lookout provides a full client-server networking service through the use of NetDDE. Use the Lookout NetDDE service to link nodes as servers, clients, or in a peer-to-peer configuration. With this service you can monitor and control your process from multiple workstations on a network. See Chapter 14, *Networking*, for information on networking Lookout computers.

Redundancy Service

Use the Lookout redundancy service to configure two computers for redundancy, providing automatic transfer of monitoring and control should one of the computers fail. See Chapter 15, *Redundancy*, for information on configuring computer redundancy.

3

Getting Started

This chapter explains how to start and get around within Lookout. It describes the Lookout screen and introduces some important mouse and keyboard shortcuts.

Starting Lookout

To Start Lookout

Note ***If you are using Lookout outside of the United States, you may be required to use a hardware key. Be sure to plug the key into the parallel port on your computer before activating Lookout. If you do not, the program will not run.***

Launch Lookout by selecting **Start»Programs»Lookout**, or double-click on the Lookout icon.

The first time you start Lookout, you should see the following display.

At this point, Lookout does not have a process file open. No monitoring and control are taking place.

To Open a Process File

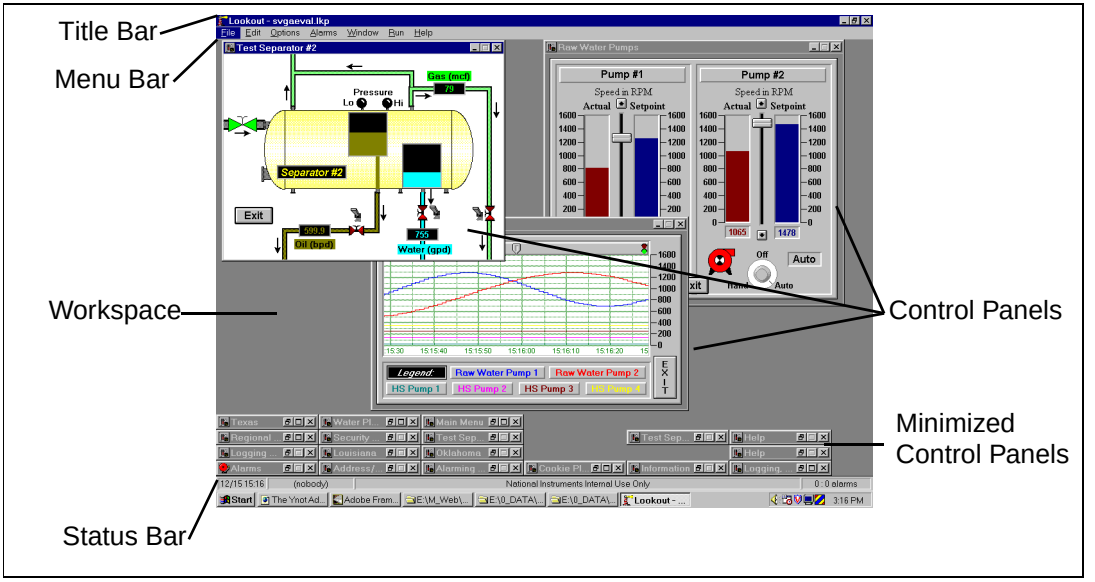
1. If you have a Lookout process file available, select the **File»Open…** menu item from the Lookout menu bar.
2. In the **File name** data field, enter the name of a Lookout process file, or choose a process file from the list and select **OK**.

Lookout process files have a `.LKP` file extension. If Lookout finds a valid process file, Lookout opens the file and immediately begins executing the process.

When a process file opens, Lookout makes additional menu selections available and displays control panels and the alarm window.

The Lookout Screen

Lookout first appears on your screen maximized, taking up the entire screen.



The Title Bar

The title bar at the top of the Lookout window displays the program name and the title of the currently executing process. If a control panel is maximized, the control panel name appears in the title bar.

The Menu Bar

The menu bar displays the currently available menu commands. These commands may or may not be available to the operator, depending on his or her security level.

The Status Bar

The status bar is at the bottom of the Lookout window. It changes when toggled into and out of edit mode. When not in edit mode, the status bar is gray and the time and date are displayed on the left end of the bar. The account name of the currently logged on operator comes next. The company name as entered during registration appears in the middle, and the alarm status is on the right end of the status bar.

When in edit mode the status bar turns yellow. The X and Y coordinates of the currently selected item appear on the left end of the bar. The width and height of the currently selected item come next. The tagname, filename, or expression of the selected item appears in the middle, and the alarm status appears on the right end of the status bar.

To toggle in and out of edit mode, select **Edit»Edit Mode**, or press <CTRL-Space>

The Lookout Workspace

The Lookout workspace is the area between the menu bar and the status bar. The workspace is the area in which you view and operate control panels. The alarm window also appears within the workspace. You organize and arranged control panels in this workspace area.

The visible workspace on your screen is only a window into the Lookout virtual workspace. If control panels or their associated icons are partially or completely outside the visible workspace, Lookout automatically displays horizontal and vertical scroll bars along the right side and bottom of the visible workspace. If scroll bars are visible, you can scroll around in the virtual workspace to see all of your control panels.

Control Panels

Control panels consist of switches, knobs, bar graphs, digital displays, trend graphs, and other components that you can use to visually monitor and control your operations. Panels can be full-screen, minimized, or any size in between. You can move the panels around the screen by grabbing the title bar of a panel with the mouse cursor and dragging it to a new location. Control panels can also “pop up” when an event occurs such as when a pushbutton is pressed or when an alarm is activated. There is no limit on the number of control panels you can create or the number of objects displayed on any one panel.

There are two types of control panels, Normal and Popup. A Normal control panel can be maximized, normal size, or minimized within the Lookout workspace. A Popup control panel in either a Popup state or minimized. When a Popup control panel is popped up, it remains on top of all other panels until you minimize it.

Control panels do not have a standard appearance. You can customize each control panel for your particular control strategy with switches, potentiometers, and other graphical objects. Some control panels may only display information, while others combine control and display information. The information displayed always represents the most current values available to Lookout through its communications with your I/O devices.

With Lookout you have considerable latitude in how you make control panels look and operate. For example, you can draw your own switches and substitute them in place of the standard switches. You can easily make a pushbutton or switch work backwards from the way one might ordinarily expect. You might insert time delays and conditions behind the panel that may not be obvious just from viewing the panel.

Control panels are windows into your process you use to control equipment by flipping switches, pressing buttons and turning knobs. Unlike real switches and knobs, you can assign security levels to individual controls, as well as entire control panels within Lookout. You can use this feature to lock out operators that do not have high enough security levels for specific devices. See Chapter 10, *Security*, for more information on Security.

Unlike physical switches and knobs, you can display the same control object multiple times, both on the same control panel and on different control panels. For example, when you turn a knob in one location, all copies of that knob turn at the same time, in each location of that control. The same control object may appear differently in each location. You might

display a potentiometer as a horizontal slide on one panel, a pair of increment/decrement buttons on another panel, and a knob on yet another panel.

Operator Input

There are numerous methods for plant operators to make setpoint adjustments or switch from one control panel to another. These include mouse, trackball, touchscreen, and keyboard activated commands.

When the cursor moves over a controllable object, the cursor turns into a hand, indicating you now have control of the object. Controllable objects include such things as switches, pots and pushbuttons.

When using a keyboard, the arrow keys move the cursor around the screen. The <Tab> key jumps the cursor from one controllable object to another, and the <Space> bar acts as the left mouse button, so you can click on a controllable object without actually using a mouse.

You can also tie keyboard function keys to control panels and other objects, so you can switch between control panels or issue control commands just by pressing a function key.

Virtual Keypad

When you are in operating mode, you can click on a digital pot control and bring up a virtual numeric keypad to enter numeric values, either with a mouse or a touch sensitive screen.

Virtual Keyboard

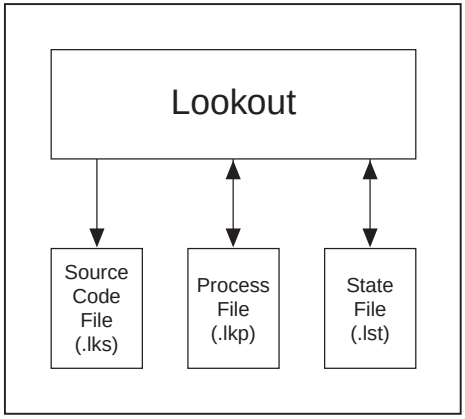
Lookout also has a virtual keyboard you can use with a touch sensitive screen or a mouse.

To enable the Virtual Keyboard, select **Options»System…** and then check **Left Mouse Click** or **Right Mouse Click** in the Virtual Keyboard Pops Up section of the dialog box.

When this feature is enabled, clicking in a data entry field or touching the field on a touch sensitive screen displays the Virtual Keyboard.

The Development Environment

With the Lookout architecture, you can quickly develop a complete process file. All development is interactive. You do not have to program or compile. Just create objects and connect their signals. The development cycle creates three unique files for every application: a process file, a source code file, and a state file.



The Process File

A *process file* is a compiled executable file that contains complete configuration information for your particular process or application. It is updated every time you select the **File»Save** command. Process files are binary files with an `.LKP` (Lookout Process) file name extension. You do not directly edit or modify process files with a text editor. Rather, Lookout compiles the `.LKP` file on an object-by-object basis as you create each object. With this approach you can stay on-line while creating, updating, or editing your application. There is no need to shut down your process. Lookout does not cease operation while you add or delete controllers, I/O, graphics, or modify your logic.

The Source Code File

Lookout also automatically updates a corresponding *source code file* when you invoke the **File»Save** command. This file contains complete documentation for your process file, including object definitions, tagnames, I/O configuration, communications, control logic, control panel layout and other object parameters. Source code files have an `.LKS` (Lookout Source) file name extension. These are standard ASCII text files

that you can print or view with any word processor or text editor. You can use this information for debugging and documentation purposes.

The State File

The *state file* contains the actual values of object data members. These values include setpoints and other important data held within and used by objects. You might think of the state file as permanent memory for Lookout setpoints and real-time trends. Lookout uses the state file to store the position of switches, potentiometers, and trend lines. When Lookout first loads an application, it reads the appropriate state file to determine what state the pots, switches, trend lines, and so on should be in upon startup.

☞ **Note** ***Hypertrends do not use the state file. Hypertrends use the Citadel database.***

The state file updates any time you save, close or exit the Lookout application file. It can also update on a periodic basis as defined in the System Options dialog box. You invoke this dialog box by selecting the **Options»System…** menu command. State files have an `.LST` (Lookout State) file name extension.

The Development Process

The first step in developing a process file is creating a control panel object. Control panels are windows you use to place other objects, such as switches, pots, and trends to be displayed. You can make them look like physical control panels, complete with switches, pushbuttons, bezels, insets, lamps, gauges, and so on. Your imagination is the only limit to how the control panels look and function.

If you select the **File»New…** command, it automatically invokes the dialog box for creating the first control panel object.

After you create at least one control panel, you can then create and display any number of other objects. Remember that objects can be displayed on a single control panel or on multiple control panels. For example, you can insert the same switch on several panels. If you flip the switch on one panel it immediately flips on all the other panels.

The first objects you might want to create could be those that represent your PLCs and RTUs. These object classes often use your computer communication ports, so you may have to configure your ports for radio, dial-up modem, or hard-wired connection, as appropriate. See Chapter 6, *Serial Communications*, for information on configuring communications.

When you create an object that represents a PLC, RTU, or other I/O device, its associated functionality and database are part of the object. The functionality built into this type of object class includes a protocol driver, enabling the object to talk with the physical device. The native (default) database includes all registers, bits, I/O signals, parameters and other values that the physical device can transmit to or receive from Lookout.

At this point, you may want to add data members to the native databases for your objects. Such new data members, called aliases, can include descriptive names, signal scaling parameters, alarming parameters, and so on.

As you create objects, you can connect them to other objects. The type of objects you create (such as Pot, Switch, Modbus, Trend, and so on) and the manner in which you connect them determine how your system interacts with your process.

Many object classes you use take advantage of the Environment Services that Lookout provides. For example, you can assign a security level to the Pot object class, and the Alarm object class uses the Alarm processing subsystem. As you create objects that take advantage of these services, you should configure the services to custom fit your application requirements.

Invoke the **File»Save** command to save your logic and graphic configuration to disk, and creates or updates the `.LKP`, `.LKS` and `.LST` files.

It is a good idea to save often while working, to avoid inadvertent loss of work. It is also a good practice to back up your work to a separate disk, regularly.

Configuration Shortcuts

Mouse Shortcuts

This manual references a number of mouse-implemented shortcuts. Becoming familiar with mouse commands is easy and pays significant dividends during application development.

Function	Mouse Action
To **select** object display, graphic, text, expression, or any item on a display panel	**Click**: Select the item by pointing the cursor at the item and clicking the left mouse button. (*See Note*.)
To **create** a new object from an existing object	**CTRL-Drag**: Select the item(s) you want more of. Hold the CTRL key down and drag the selected item(s). This makes a completely new instance of the control: <CTRL>-Dragging Pot1 will produce a new object named Pot2.
To **copy** object display, graphic, text, expression, or any item on a display panel	**Shift-Drag**: Select the item(s) to be copied. Hold the Shift key down and click the left mouse button while dragging the item(s). This makes a duplicate of the selected object that is linked to the original. Both the original and the duplicate have the same tag name.
To **modify display or definition parameters** of an object, graphic, text, expression, or any item on a display panel	**Right-Click**: Select the item(s) to modify. Position the cursor over the item(s) and click the right mouse button.

☞ **Note** ***You can select multiple items on a display panel by clicking in a panel and dragging the box outline around the objects you want to work with. Subsequent commands affect the entire group at once or toggle from one item to the next, prompting you for new instructions. You can stop this process at any time by holding the*** <Shift> ***key down and selecting either the*** **OK** ***or Cancel buttons in the current dialog box.***

Remembering Tagnames

Many dialog boxes contain *expression fields*. Expression fields are yellow to indicate that you can get help identifying or remembering a tagname and associated data members by pressing the right mouse button (right-clicking on the selected expression field).

Assume, for example, that you want to trend the flow in a return water line. You start to create the trend object, but you cannot remember the tagname

you want to trend. Right-click on the trend line expression field (the field in which you normally type the tagname to trend).

When you right-click any expression field, Lookout presents the Expression Editor dialog box, as shown in the following diagram.

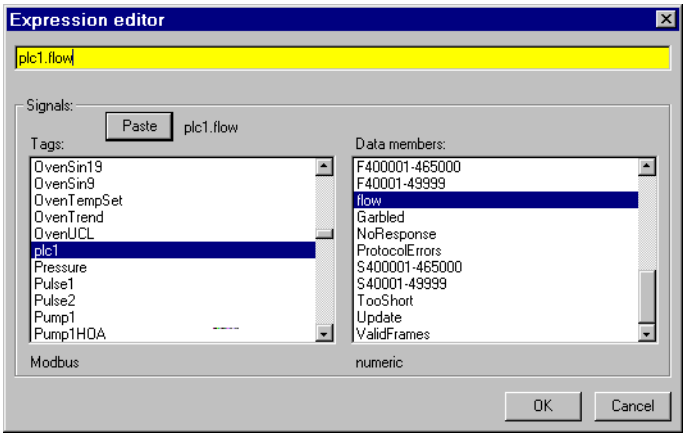
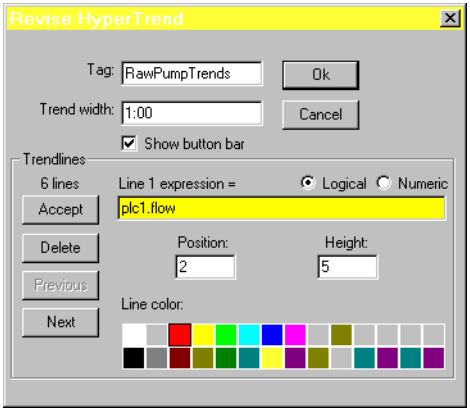


Figure 3-1. The Expression Editor Activated by Right-clicking on a Yellow Data Field

Using the Expression Editor dialog box, you can easily paste object names and their data members directly into the expression field. After you select an object and a data member, paste them into the expression editor data field, and click on **OK**, Lookout writes your expression into the trend expression data field.



4

Using Lookout

This chapter discusses the basics of using Lookout, including selecting objects, creating objects, editing object databases, and other operations.

If you have not read the discussion of Lookout architecture in Chapter 2, *Introduction*, and the discussion of basics in Chapter 3, *Getting Started*, you should go back and read those chapters. This chapter builds on information presented in the earlier chapters, and assumes that you have absorbed that information

The keys to using Lookout are objects—their parameters and data members. The technique for using Lookout involves solving problems by selecting objects and making connections.

Selecting Objects

How do you decide what objects to pick and what connections to make?

Some decisions are easy—you will certainly use driver objects for the different PLCs, RTUs and other devices you need to monitor and control. Other decisions are driven by what you want to do.

To solve problems with Lookout, ask yourself which object does what you need done.

Lookout has many objects with obvious uses, such as pushbuttons, switches, and trend displays. Other Lookout objects may not be as immediately obvious as to how they might be used. To familiarize yourself with Lookout objects, you should browse through Chapter 18, *Object Class Reference*.

After you become familiar with the Lookout objects, you have a better idea of which object to select—or which objects to investigate—to solve the problem you are facing at the moment.

Creating Objects

To Create an Object

Complete the following steps to create a Lookout object.

1. Make sure you are in Edit Mode. (Select **Edit»Edit Mode** if necessary.)
2. From the Lookout menu bar, select the **Object»Create…** command. For this example, select the **Pot** class of objects, found in the **Control** category.

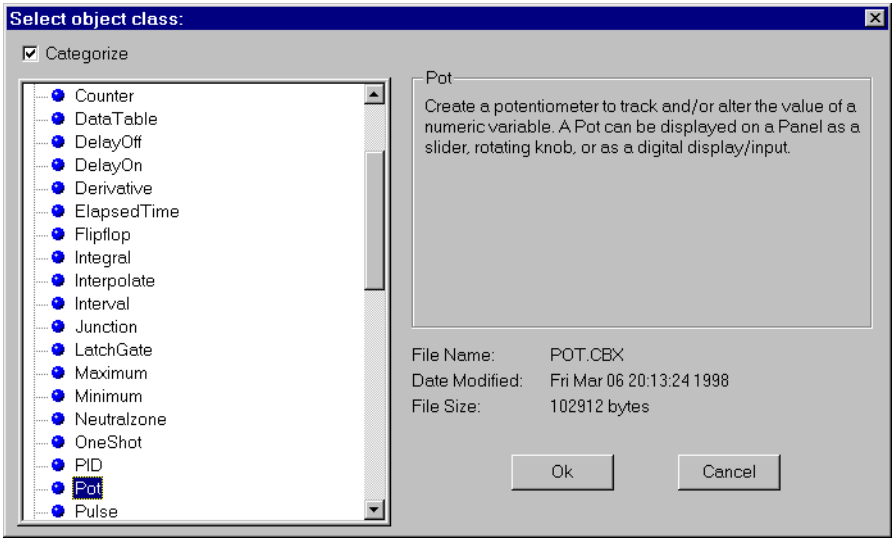


Figure 4-1. The Select Object Class list box

When you select an object class, a dialog box you use to define the object parameters appears.

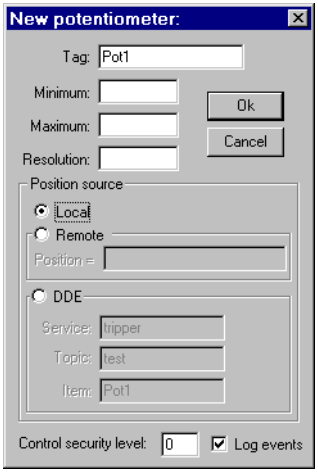


Figure 4-2. An Object Definition Dialog Box

3. Assign a unique tagname to the object. Valid tags are described in *Object Tagnames*, below.
4. Define the object parameters. Each object class has unique parameters, described in Chapter 18, *Object Class Reference*.

   Some parameters can be defined as *expressions*—roughly the equivalent of spreadsheet formulas—that you can use to manipulate data. See Chapter 7, *Expressions*.
5. Select **OK** to create the new object.

   Many object classes are not displayable on a control panel, such as a protocol driver object for a typical PLC, or an object such as Neutralzone or Pulse (both Lookout objects with special functionality).

   Other object classes, like Pots, are displayed on a control panel. When you create such an object, Lookout presents you with a Display dialog box you use to define how the object is displayed.

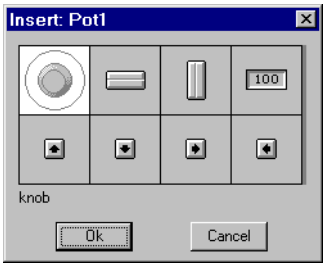


Figure 4-3. Pot Display Dialog Box

6. In the Display dialog box, choose how you want the object to be represented and select **OK**.
7. Position the object on the control panel and adjust the size.

☞ **Note** ***You can copy objects to paste onto another Lookout panel, but if the panels are of different sizes, the pasted objects may not appear in the proper location on the new panel.***

Object Tagnames

A tagname is the name that you give to a specific object. Tagnames include the characters A – Z, a – z, 0 – 9, and the underscore character (_). Tagnames must begin with a letter, and can be up to thirty two (32) characters long. Spaces and tabs are not allowed.

Lookout recognizes uppercase and lowercase characters as the same character, so SW2 and sw2 are the same tagname.

You must assign each object a unique tagname. Lookout does not allow you to create or modify an object if the tagname is not unique or if it does not follow the proper tag naming convention.

Examples of Valid Tagnames
Pump3_HOA
HighServicePump3_Run
ClearWellLvl_at_Hiway289_and_I35
ShaftTempF
Pushbutton1
runtime
Sw1
Sw2

Using Lookout keywords (object class names and function names) as tagnames is not allowed. For example, you cannot use terms such as Switch, DelayOn, STDEV, Modbus, nif, and Trend.

Invalid Tagnames	Reason
3Pump_HOA	begins with number
_HighServicePump3_Run	begins with underscore)
ClearWellLvl at Hiway289 and I35	contains spaces
#ShaftTempF	begins with #
Pushbutton	reserved word
ThisTagIsWellOverThirtyTwoCharacters	too long
and	reserved word

Editing Object Databases

Lookout creates the *native* database of an object automatically when you create the object. The native database is documented at the end of each object class definition in Chapter 18, *Object Class Reference*.

You can create new data members or modify the parameters of any existing native data member. These parameters include such things as alarm setpoints, deviation filters, scaling factors, historical logging, and alias names.

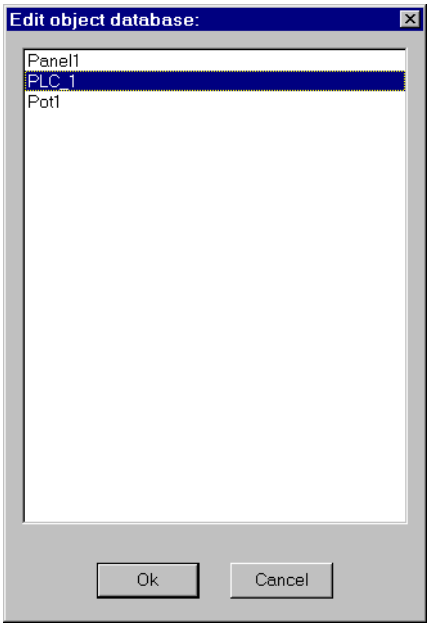
☞ **Note** ***Any object in Lookout can have its native database modified. However, this is most practical for objects with large native databases, such as driver objects and Data Table objects.***

To Edit Database Parameters

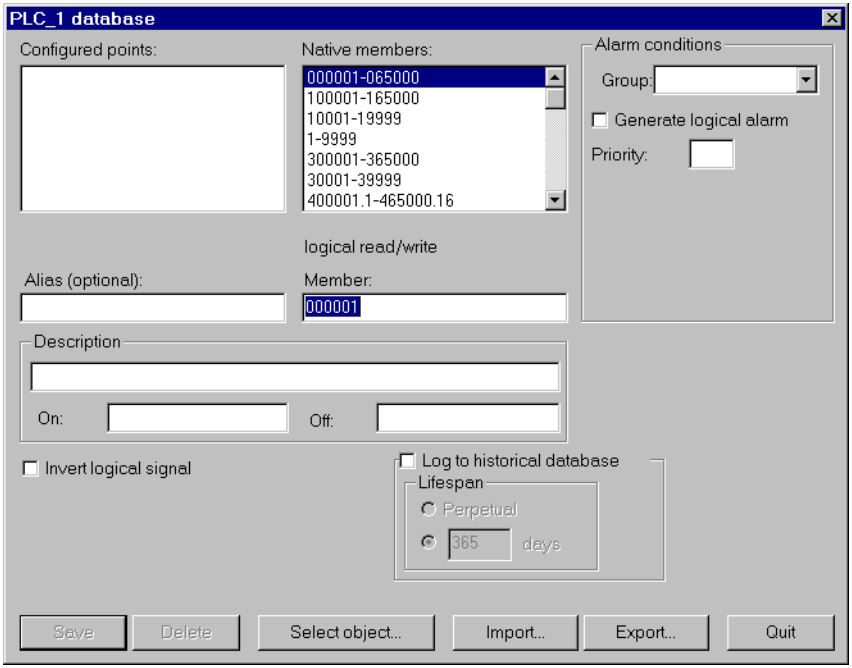
1. For the purposes of this example, create a Modbus object. Use the tagname PLC_1.

   The Modbus object can be added to your system without a physical connection while you are learning to use Lookout. Let the default settings stand when you create the object.

2. From the Lookout menu bar, select **Object»Edit Database…**.



3. In the Edit object database list box, choose PLC_1.
   The following dialog box appears.



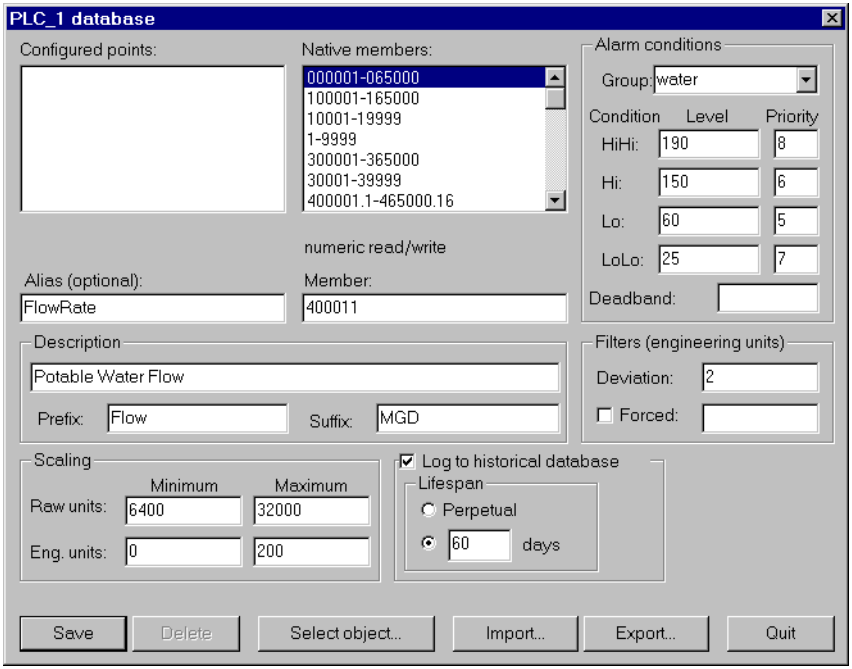
4. For each data member to be configured:

   a. Identify the desired data member by entering it into the **Member** data field. If you are modifying a data member that has been previously configured, you can select it from the **Configured points** list box.

      For this example, select `40011` as the modbus data member. Enter `FlowRate` as the alias

Note ***Parameter fields automatically change depending on the data member you select. Lookout automatically determines whether the data member is logical or numeric, and presents you with the appropriate parameter attributes.***

   b. Configure appropriate parameters shown in the following illustration. See the individual parameter sections in this chapter for details on setting each parameter.



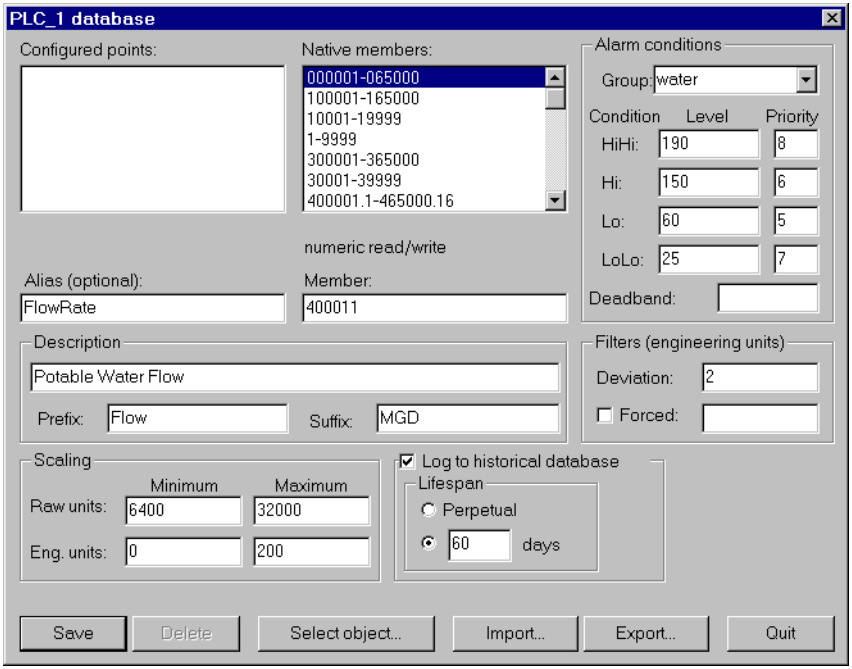
c. Select **Save** or **Update**. (If you are modifying a data member that was previously defined, the **Save** button changes to an **Update** button.)

Lookout stores all the new parameter settings for the specified data member when you select **Save** or **Update**. In addition, Lookout adds the modified data member to the **Configured points** list box for future reference. Lookout immediately reflects these changes throughout your configuration.

5. Select **Quit** to exit the dialog box.

Numeric Member Parameters

The following diagram and paragraphs describe numeric data member parameters. Logical data members, covered in the following section, share a number of the same parameters.



Alias renames any native data member. You can think of an alias as a sort of nickname. For example, the Modbus driver object includes the native data member `40011` that represents an analog input. You can give this native data member an alias like `FlowRate`. From then on, you can reference the alias `FlowRate` instead of its native name `40011`. All associated parameters (such as **Scaling**) are also applied to the alias value.

An alias is a good way to insulate your Lookout configuration from changes in your PLC, RTU, or I/O configuration. For example, consider a flow transmitter wired to an analog input at `40011`. You can give `40011` the alias name `FlowRate`, just as you did in the example. Multiple control panels can then display the `FlowRate` data member and numerous other objects in Lookout can use it. If you later rewire the transmitter to the analog input at `40012`, you need only modify the alias `FlowRate` to reflect the new I/O address. Lookout instantly reflects this change everywhere `FlowRate` is used.

You can modify all associated parameters of an existing alias *except* the alias name itself. If you attempt to modify an existing alias name, the **Update** button changes to a **Save** button and you will only create a new alias.

Most developers implement aliases on objects with large native databases, such as driver objects (like Modbus and Tiway) and Data Table objects.

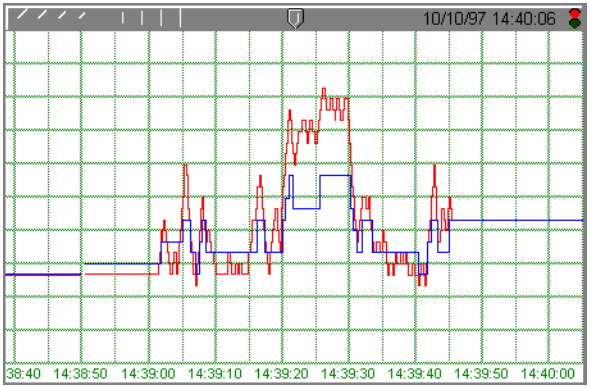
Note ***An alias is optional. You can apply scaling, alarming, and logging parameters to any native data member and save it to the*** *Configured points* ***list without giving it an alias name.***

The **Description** appears as the message text in the alarm window. It can have spaces, and it can be lengthy. You do not have to enter quotes in this field.

Prefix and **Suffix** are part of the description, but do not appear in the alarm window. They are just additional descriptive text.

Define **Scaling** by entering **Raw units** and **Eng. units**. The raw numeric data member is converted (scaled) to an engineering unit value. The PLC in this example generates a raw value ranging from 6,400 to 32,000. Lookout converts that raw signal to range from 0 mgd to 200 mgd. The conversion is linear. See your hardware specifications and calibration records for the minimum and maximum raw units associated with analog devices. If you leave the **Raw units** and **Eng. units** fields blank, Lookout performs no scaling on the signal.

Deviation filters out insignificant variations of numeric signals. The following figure shows two values plotted on a trend. One line is the raw unfiltered value. The other, stair-stepped line, represents the filtered value after passing through a **Deviation** of 2.



The Citadel database also uses **Deviation** as the criteria which triggers logging of new historical data to disk. See Chapter 11, *Logging Data and Events*, for more information on logging data.

Use the **Forced** data field to manually enter a constant value for the data member. When you select the **Forced** check box and enter a value in the field, Lookout forces the engineering unit value to be equal to the value you entered—regardless of the actual value of the native data member. You might use this when a sensor fails or during sensor maintenance, or any time a PLC receives a bad signal from the transmitter.

Use the **Alarm condition** parameters to define alarm limits and their associated priorities. Lookout compares the alarm setpoints to the engineering units value (that is, the post-scaled, post-filtered number). If you do not enter scaling parameters, Lookout applies the alarm parameters directly to the raw signal.

You can assign an alias or native data member to any existing alarm **Group** or you can create a new **Group**. To create a new **Group**, enter the new group name in the field. See Chapter 9, *Alarms*, for more information on alarms.

Use the alarm **Deadband** parameter to prevent fluttering between alarm and normal states when the signal value hovers near an alarm limit. The following figure shows a value plotted against its **Hi** and **Lo** alarm setpoints.

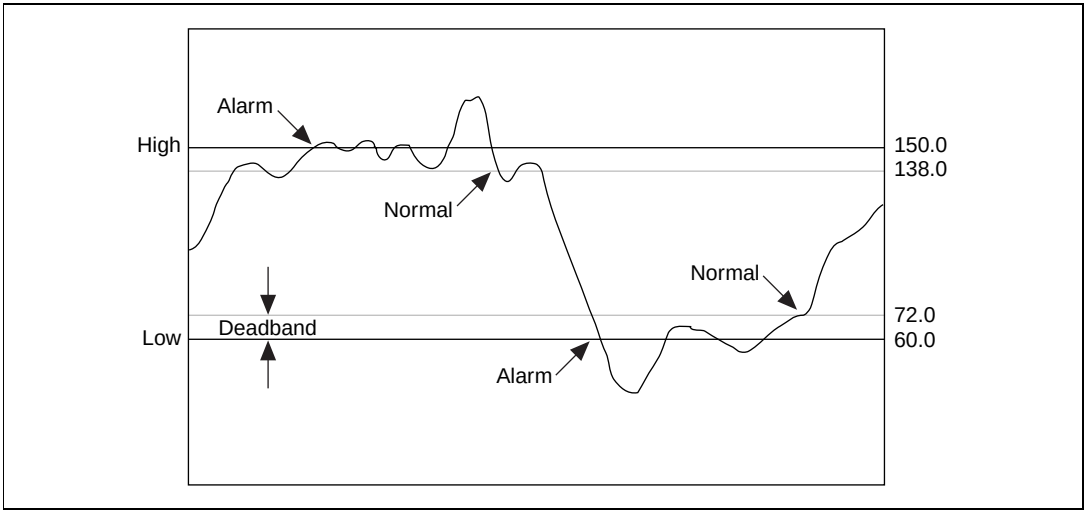


Figure 4-4. Alarm processing of a data member whose alarm setpoints are shown and whose **Deadband** is 12

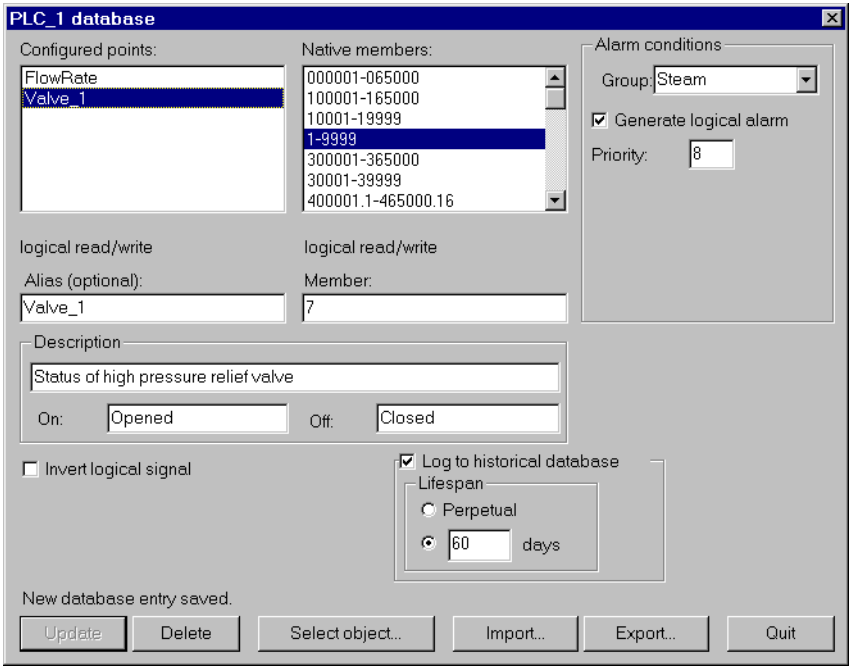
Lookout generates an alarm the moment the value violates the **Hi** or **Lo** alarm setpoints. The alarm returns to normal when the value drops below the high alarm setpoint minus the Deadband, or goes above the low alarm setpoint plus the Deadband. The Deadband also applies to all **HiHi** and **LoLo** alarm limit setpoints.

The **Log to historical database** parameters define how long to store a value in the Citadel database on your hard drive. If you do not select this option, Lookout does not log the value to your disk. If any scaling or filtering parameters are defined, Lookout logs the scaled, filtered value (that is, the engineering unit value). See Chapter 11, *Logging Data and Events*, for more information on logging data.

Logical Member Parameters

Some of the parameters of logical data members are different from those of numeric data members. Scaling of a logical signal consists of the **Invert Logical Signal** check box. When you choose this check box, an ON value is represented by an OFF value, and so on. When you do not select it, ON is ON and OFF is OFF.

A logical database signal is shown in the following figure.



Alarm parameters of a logical signal include the alarm **Group** assignment field and the **Generate Logical Alarm** check box. When selected, the data member generates an alarm whenever the value is ON; the alarm condition clears whenever the value is OFF. Notice that if the Invert Logical Signal check box is selected, the value used here is the inverted value. See Chapter 9, *Alarms*, for more information about alarms.

Text Member Parameters

The text data member database contains only **Alias**, **Member**, and **Description** fields.

Importing and Exporting Object Databases

Use the import database service to copy database member parameters from an Excel spreadsheet file directly into an object. Use the export database service to copy an object database into an Excel spreadsheet file. This is what you can do using these services:

- Export object database parameter definitions to Excel for the purpose of documentation.
- Export an object database to Excel, perform global replacements on data member parameters, and then import the changes.
- Create a tag list in Excel or in an application that exports to Excel, then copy that tag list into Lookout.
- In a process using multiple duplicate driver objects (such as a gas pipeline or water distribution system), define a single driver object database parameters in Excel. Import that database into multiple driver objects.

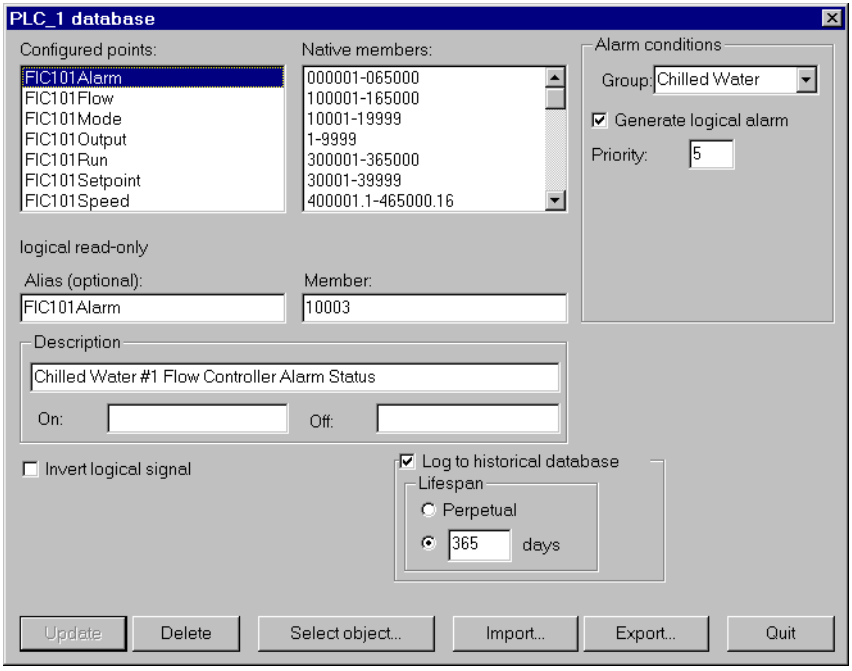
Although you can import and export any object database, you may find that these services are most useful for objects with large native databases, such as driver objects and Data Table objects.

Exporting an Object Database

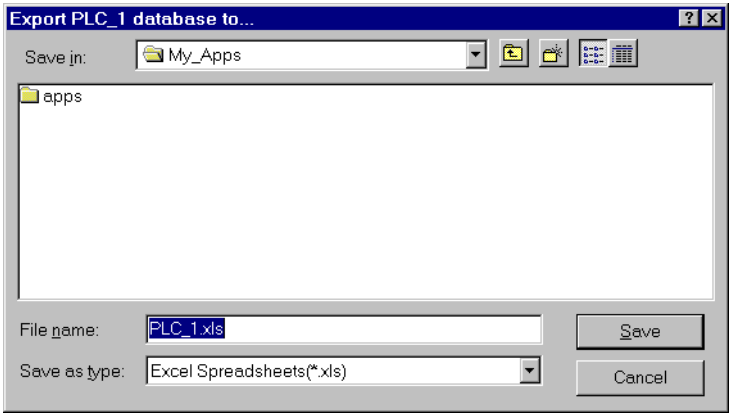
To export an object database:

1. From the Lookout menu bar, select **Object»Edit Database…**.
2. In the Edit object list box, choose the object you want.

   The following diagram shows a Modbus object that already has a number of logical and numeric data members defined.



3. In the database dialog box, click on the **Export** button.

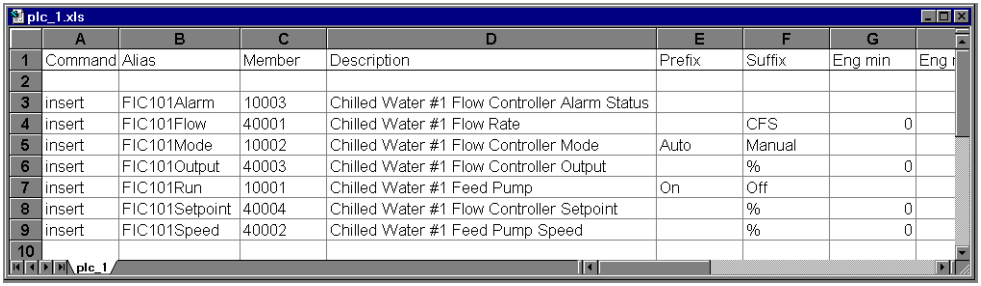


4. In the Export Object Database to… dialog box, choose a directory path, enter a filename and click on **OK**.

When you export a database, Lookout does not export every possible data member (many driver objects have a capacity for thousands of members). Instead, Lookout exports configured points; that is, data members that have

at least one parameter already defined. Lookout also exports native members that are in use (that is, connected to other objects).

The spreadsheet file that the Lookout Export command creates is in Excel Version 2.0 format. An example of the `.XLS` file is shown in the following illustration.



plc_1.xls

	A	B	C	D	E	F	G	
1	Command	Alias	Member	Description	Prefix	Suffix	Eng min	Eng
2								
3	insert	FIC101Alarm	10003	Chilled Water #1 Flow Controller Alarm Status				
4	insert	FIC101Flow	40001	Chilled Water #1 Flow Rate		CFS	0	
5	insert	FIC101Mode	10002	Chilled Water #1 Flow Controller Mode	Auto	Manual		
6	insert	FIC101Output	40003	Chilled Water #1 Flow Controller Output		%	0	
7	insert	FIC101Run	10001	Chilled Water #1 Feed Pump	On	Off		
8	insert	FIC101Setpoint	40004	Chilled Water #1 Flow Controller Setpoint		%	0	
9	insert	FIC101Speed	40002	Chilled Water #1 Feed Pump Speed		%	0	
10								

plc_1

Creating a Database Spreadsheet

The easiest way to create a database spreadsheet for an object is to create the object in Lookout, define the parameters for one logical data member and one numeric data member, and then export the database. The export function automatically creates all the necessary column labels, and the two data members furnish examples you can follow.

Notice that Row 1 contains column labels. These include the names of all possible data member parameters. Lookout references the labels, not the column numbers, so you must spell all column labels exactly. Lookout ignores white space and is case-insensitive. The table below lists all possible column labels.

Table 4-1. Spreadsheet Column Labels

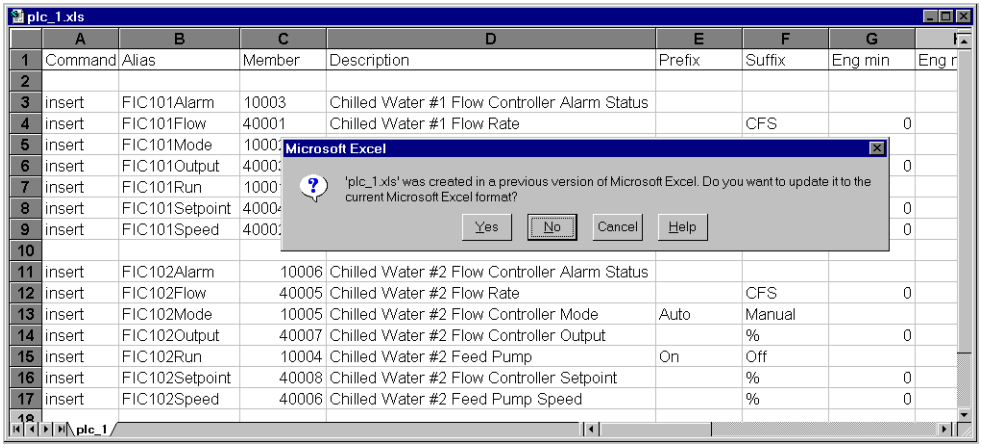
Command	Member	Alias	Description
Prefix	Suffix	Eng min	Eng max
Raw min	Raw max	Invert?	Deviation
Force?	Forced value	Alarm group	Lolo level
Lolo priority	Lo level	Lo priority	Hi level
Hi priority	Hihi level	Hihi priority	Logical priority
Log data?	Lifespan	*(Empty if not assigned)*	*(Empty if not assigned)*

Lookout requires the column labels **Command** and **Member**. All other column labels are optional. **Command** must be in Cell A1. The order in which the other column labels appear makes no difference.

When you import a database, Lookout reads the first 30 columns (A – AD) and ignores columns that do not have labels.

The rows below the column labels (below Row 1) each represent a database member. For example, Row 5 in the spreadsheet sample above represents the data member whose alias name is FIC101Mode.

You can easily add rows to define new points. Copy the rows associated with FIC101 and then modify the new rows slightly by identifying different native members and giving them new aliases and descriptions.



	A	B	C	D	E	F	G
1	Command	Alias	Member	Description	Prefix	Suffix	Eng min
2							
3	insert	FIC101Alarm	10003	Chilled Water #1 Flow Controller Alarm Status			
4	insert	FIC101Flow	40001	Chilled Water #1 Flow Rate		CFS	0
5	insert	FIC101Mode	1000				
6	insert	FIC101Output	4000				0
7	insert	FIC101Run	1000				
8	insert	FIC101Setpoint	4000				0
9	insert	FIC101Speed	4000				0
10							
11	insert	FIC102Alarm	10006	Chilled Water #2 Flow Controller Alarm Status			
12	insert	FIC102Flow	40005	Chilled Water #2 Flow Rate		CFS	0
13	insert	FIC102Mode	10005	Chilled Water #2 Flow Controller Mode	Auto	Manual	
14	insert	FIC102Output	40007	Chilled Water #2 Flow Controller Output		%	0
15	insert	FIC102Run	10004	Chilled Water #2 Feed Pump	On	Off	
16	insert	FIC102Setpoint	40008	Chilled Water #2 Flow Controller Setpoint		%	0
17	insert	FIC102Speed	40006	Chilled Water #2 Feed Pump Speed		%	0

If you are working with a version of Excel more recent that 4.0, the program asks you if you want to update your spreadsheet to a newer format when you select **File»Save.** This dialog box is shown in the illustration above. *Be sure to select* **No**. Lookout does not currently accept Excel spreadsheet files from versions greater than 4.0.

Importing an Object Database

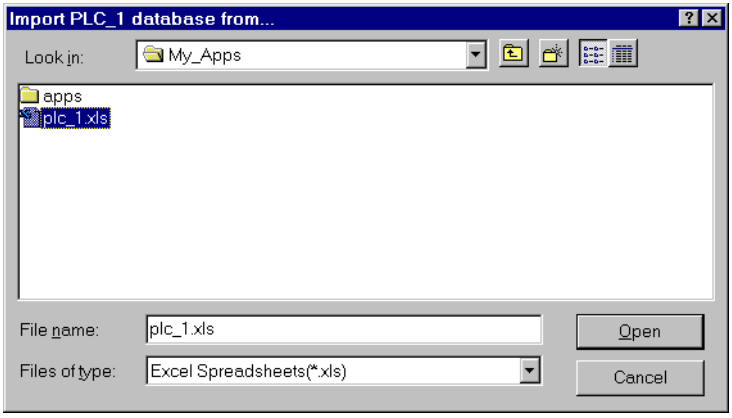
When you import a database, Lookout reads the first 30 columns (A – AD) and ignores columns that do not have labels as well as Column AE and beyond.

Each row in the Command column (Column A) contains either the keyword `insert` or the keyword `delete`. When you import a database, Lookout ignores rows that do not have the `insert` or `delete` keyword. It adds

those records whose command keyword is `insert`. It removes those records whose command keyword is `delete`. To determine exactly which records to delete, Lookout uses the record alias name; or if the record does not have an alias name, it matches the record member name.

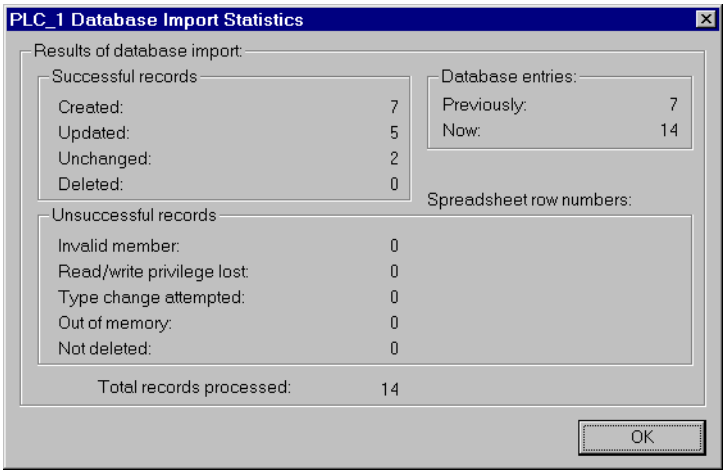
To import an object database:

1. From the Lookout menu bar, select **Object»Edit Database**.
2. In the Edit object list box, choose the object you want.
3. In the object database dialog box, click on the **Import** button.

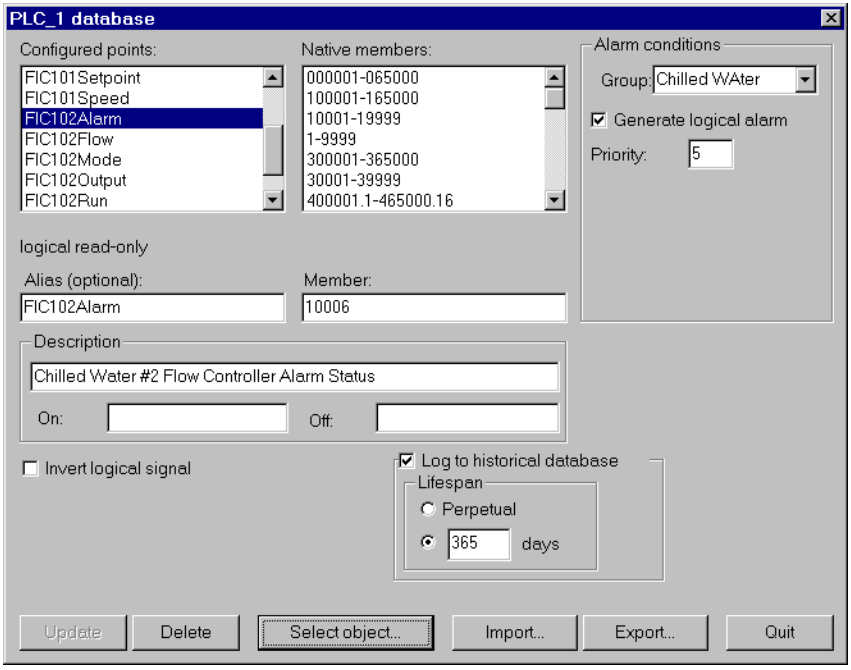


4. In the Import Object Database from… dialog box, choose a directory path, select the filename and click on **OK**.

   When finished, Lookout presents you with a set of database import statistics, as shown in the following illustration.



As you can see below, the points added to the database spreadsheet were successfully added to the **Configured points** list.



Copying an Object Database

The import and export features make it easy to copy the database of an object. This is especially useful when creating large SCADA applications, such as gas pipelines with multiple compressor stations.

The key to defining multiple driver objects that require duplicate databases is to first create an object in Lookout for each RTU or PLC. Then create a single database in Excel. Next, import that database into each driver object.

Connecting Objects

In Lookout, you connect objects to pass control signals between the objects—in much the same way you would wire a time delay relay to a motor starter relay, for instance. Once connected, objects pass signals between their various data members and parameters.

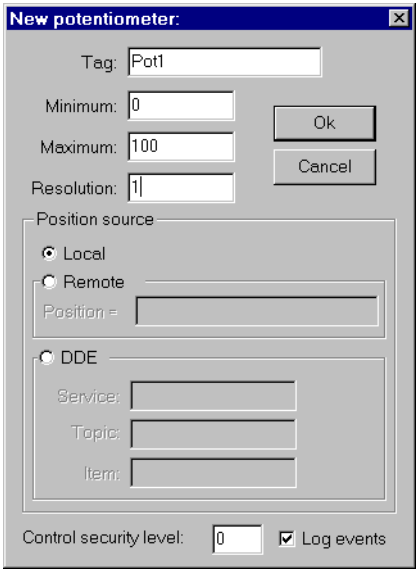
There are two basic methods to connect objects: from data member to parameter, or from data member to data member. The method you use depends on what you are trying to accomplish.

☞ **Note** ***One tip for using Lookout— when you choose an object in the*** **Edit Connections** ***dialog box, always select the object you want to connect*** **to.**

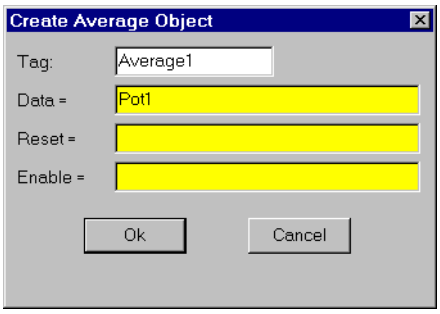
Connecting Data Members to Parameters

The following example uses the (implicit) data member of a Pot object as the **Data** source of an Average object. You can tell that **Data** is a parameter because it appears in the Average object dialog box.

1. Create a Pot object as shown. See the *Creating Objects* section earlier in this chapter for detailed information.



2. Now, create an Average object. Connect the potentiometer numeric signal to the **Data** parameter. The Average object calculates the average level of `Pot1` over time, as described in the Average object definition in Chapter 18, *Object Class Reference*.



Identifying Object Data Members

Because many objects generate multiple signals you can use as inputs to other objects, the object tagname is not always sufficient to identify the desired signal. You must identify both the object tagname and the appropriate data member. For example, a Modbus object generates thousands of signals, some logical and some numeric. To specify exactly which signal you want from a particular Modbus object, enter the object tagname followed by a period (.) and the data member you want.

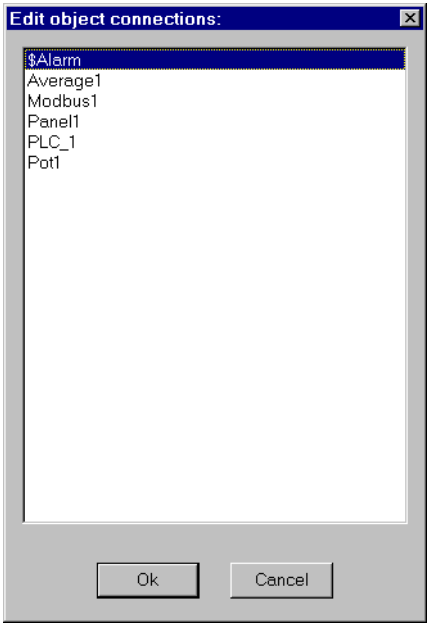
If you have a Modbus object with a tagname `PLC1`, and you want to display the value residing in holding register 40001 and the status of coil 100, reference those particular data members by typing `PLC1.40001` in one expression and `PLC1.100` in another expression.

Connecting Data Members to Data Members

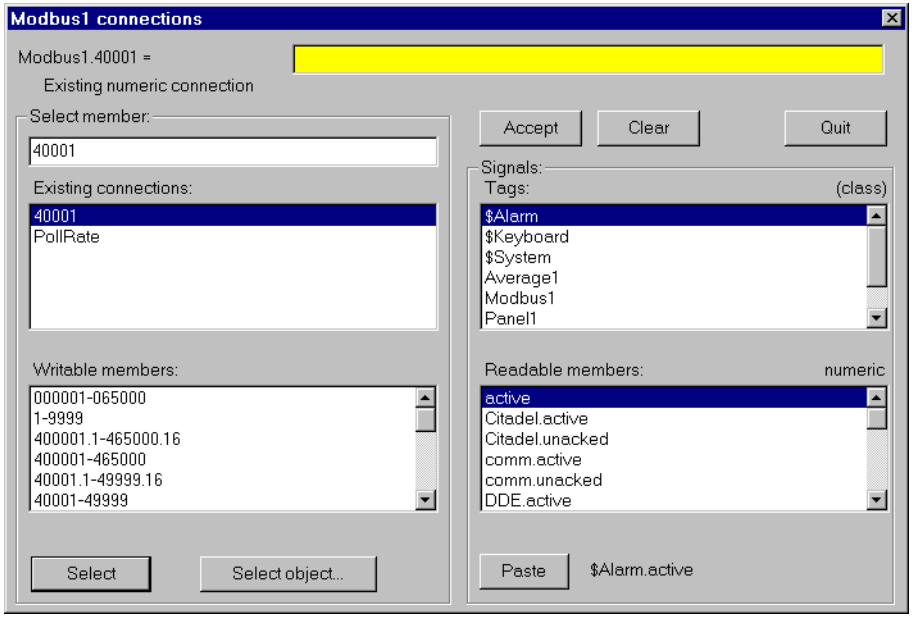
The following information expands on the previous example, in which you created a potentiometer named `Pot1`. In this example, you create a Modbus object and connect the numeric signal generated by the pot to a 16-bit holding register at address 40001 on the PLC.

1. Create a Modbus object. You do not have to have a real PLC connected to your system. Just accept the default parameter settings in the Modbus dialog box and click **OK**.
2. From the Lookout menu bar, select the **Object»Edit Connections…** menu command. You use this menu command any time you are connecting to the database of another object.
3. In the Edit Object Connections list box, choose the object that contains the data member you are *connecting to*.

   Remember, you are connecting the `Pot1` signal *to* data member 40001 in the Modbus object.

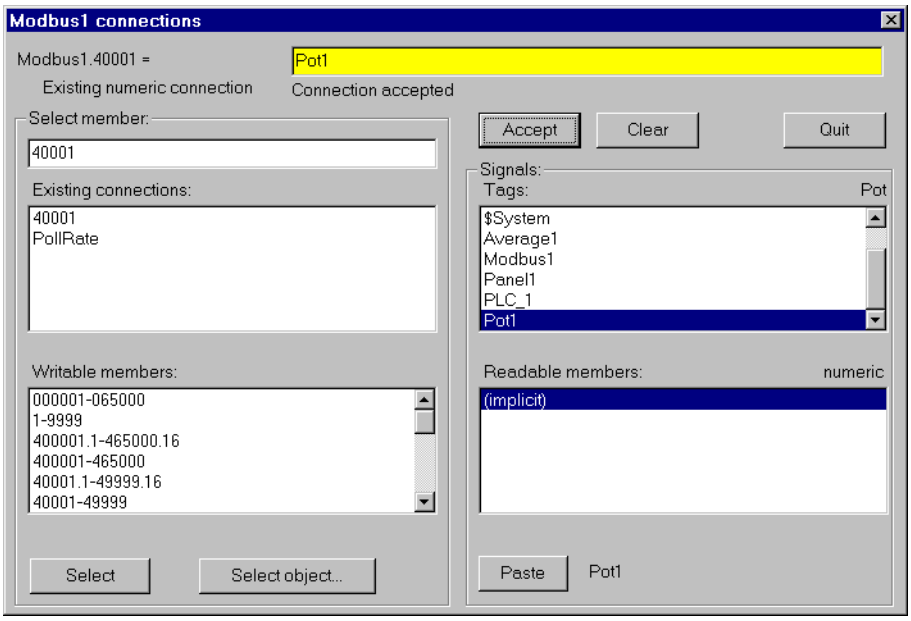


4. In the **Select member** data field, enter the data member you want to connect *to*.



Click on the **Select** button to identify the data member to be written to. Notice the dialog box now reflects the specific member; **Modbus1.40001 =**. Also notice the equal (=) sign. This prompts you to connect an expression to **Modbus1.40001**. In this case, use the expression `Pot1`.

5. In the yellow expression field, enter the tagname `Pot1`.



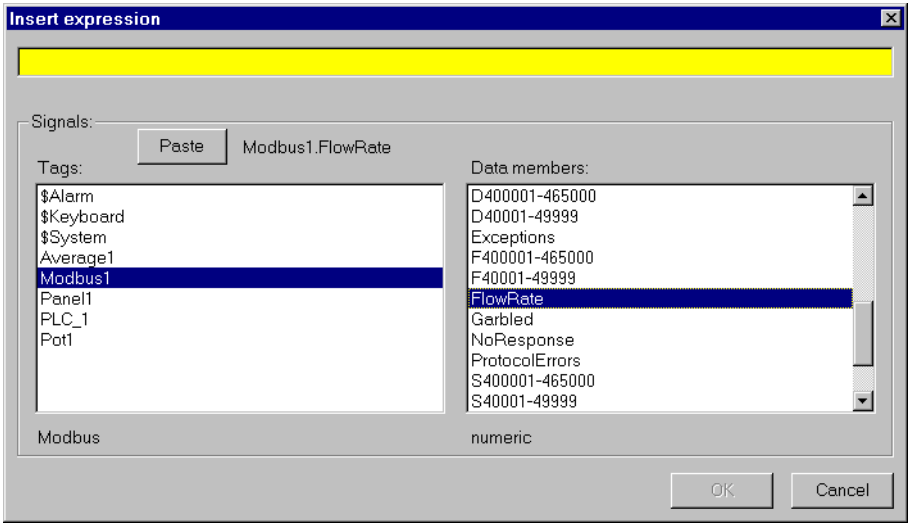
6. Click on the **Accept** button. This confirms the connection You should see a message confirming the action. You will also see data member 40001 added to the **Existing connections** list box.

You have now connected a data member of one object to a data member of another object. Any change in the (implicit) value of `Pot1` is, by definition, an event, sending a signal to all the other objects `Pot1` is connected to. Because it is now connected to `Modbus1.40001`, adjusting `Pot1` changes the value of data member `40001` in object `Modbus1`.

Displaying Data Members on Control Panels

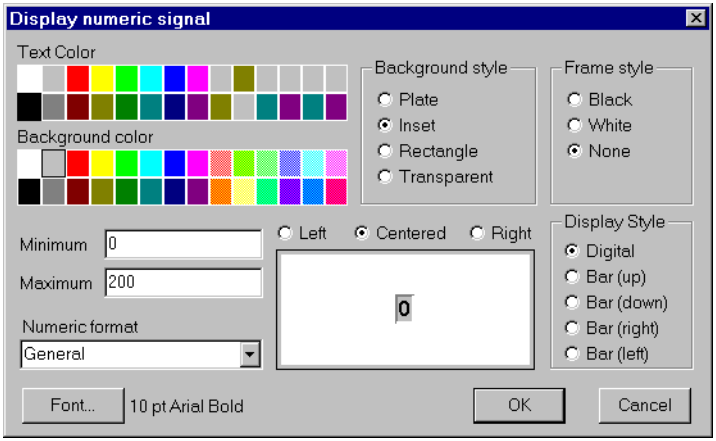
To Display a Data Member on a Control Panel

1. From the Lookout menu bar, select **Insert»Expression…**.
2. In the **Tags** list box, select the object you want, or type the tagname and data member in the yellow expression field. See Chapter 7, *Expressions*, for more information.



   In this case, select the Modbus PLC object. When you select an object, Lookout lists the object data members.
3. In the **Data members** list box, select the data member you want.
4. Click on Paste and then select **OK**.

   Lookout responds by presenting the Display Numeric (or logical) signal dialog box. The type of dialog box depends on the type of data member being inserted (numeric or logical). In this case the signal is numeric (`Modbus1.FlowRate`).



Select the display parameters you want, and click on **OK**.

5

Developer Tour

The following tour guides you through the development of a simple process file. Any time you want to learn more about a particular object class, function, or feature, refer to the appropriate section of the manual for a more detailed explanation.

☞ **Note** ***If you have not used Microsoft Windows, you should first familiarize yourself with concepts outlined in the Microsoft Windows User Guide. Learn how to use the pull-down menus, maneuver within dialog boxes and switch between Windows applications before continuing.***

Building a Lookout Monitoring System

The example system monitors the water level in an elevated tank several miles away (remote RTU), and controls a pump located in the plant (local PLC). When the tank drops below a certain level the pump kicks on to fill up the tank, when the tank is full the pump shuts off. The tank is a 50-foot tall water storage tank.

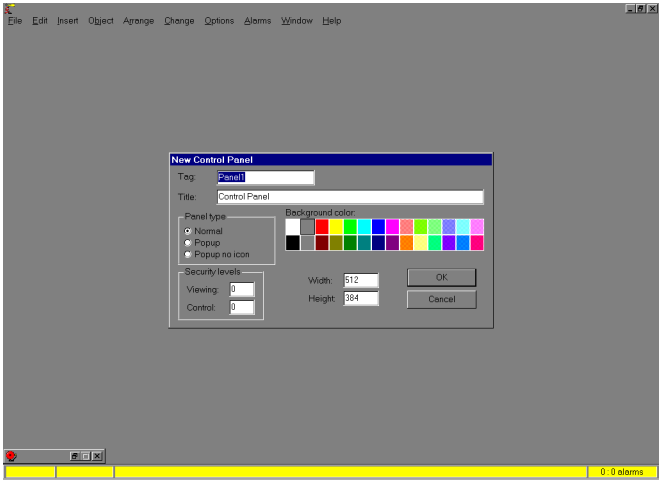
The system to be controlled by this process file consists of a PLC and an RTU, both of which communicate using the standard Modbus protocol. Modbus is the example protocol because of its widespread use in the process control and SCADA industries.

You will connect to the PLC using hardwired serial communications. You will tie the RTU to Lookout with radio communications. Assume that Lookout is at a remote radio (not the master repeater).

Create the Control Panel

Activate Lookout. It appears with a blank window.

Choose **File»New…** to begin development of your sample process file. Lookout opens a **New Control Panel** dialog box.



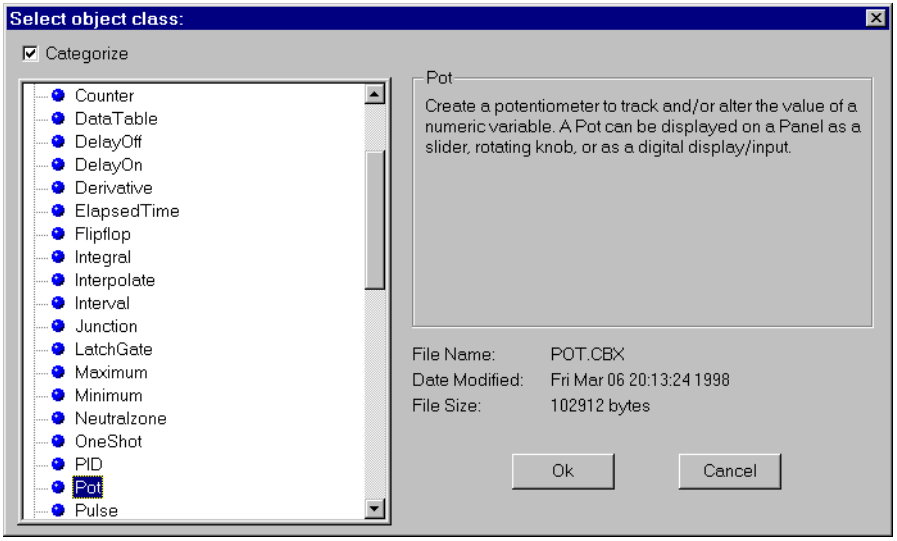
Fill in the Control Panel **Tag** name and **Title** accordingly and pick **Normal** as the panel type, and maximize the panel. It should now take up the entire workspace and the panel title appears in parenthesis at the end of Lookout title bar. You can choose not to build your control panels full screen, designing them around their normal state, instead.

☞ **Note** ***Consider screen resolution when creating display panels (for example, VGA vs. Super VGA). The same panel can appear differently on computers using different resolution display drivers. Please read about screen resolutions in the*** *Panel* ***section of Chapter 18,*** *Object Class Reference****, before designing your panels.***

Water Level Simulator

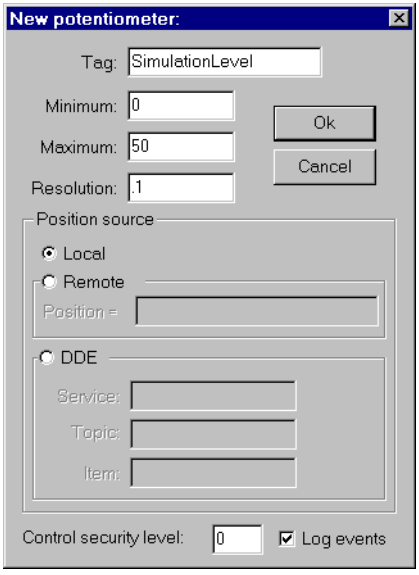
Ordinarily, you would now add the logic and graphics that control and display the pump and tank. First, however, create a potentiometer that simulates a fluctuating tank level instead of directly connecting the Modbus register for that analog input. You can use this control to simulate an analog signal if you do not have readily available Modbus devices. After demonstrating the process with a simulation pot, you can connect the system to its respective Modbus registers.

Select **Object»Create…** and pick Pot from the list of object classes in the **Control** category.

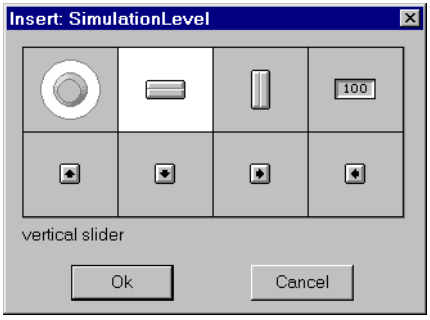


Select the object class you want and double-click on it, or select **OK**.

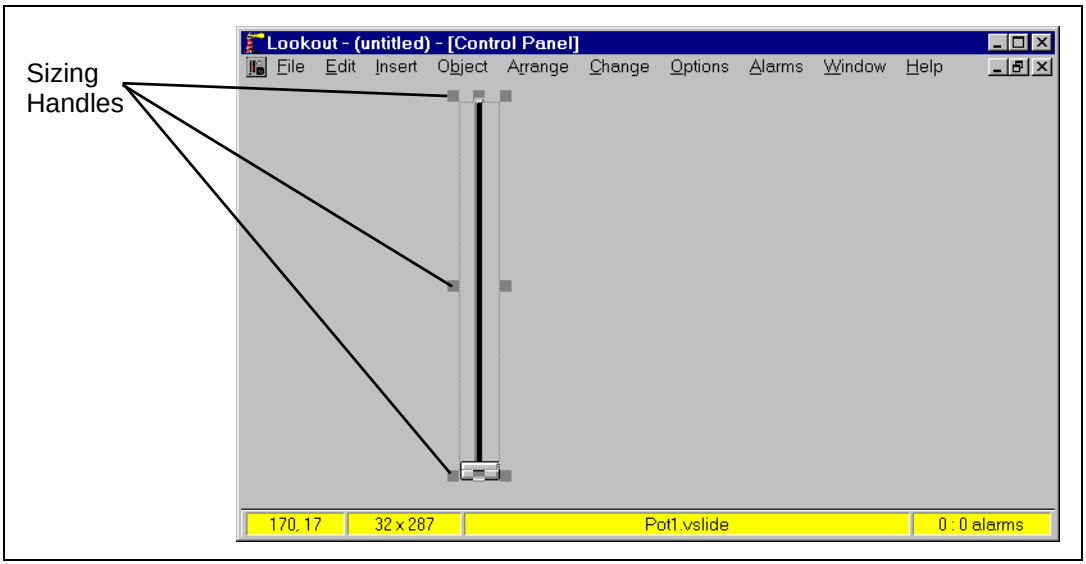
Fill in the dialog box as shown in the following illustration.



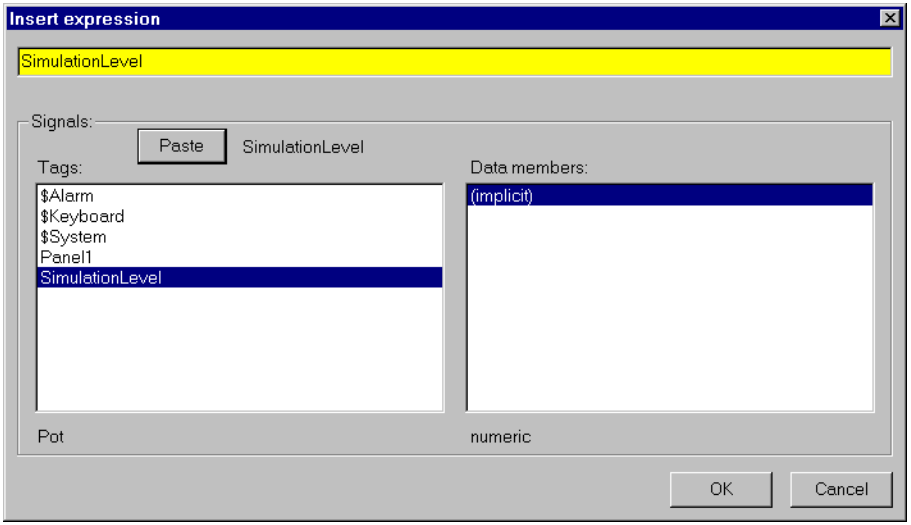
Select **OK** when you finish. Lookout immediately prompts you to select a display type for the pot.



Choose the vertical slider display and select **OK**. Lookout inserts the pot on the active control panel. You can now stretch or reposition the potentiometer anywhere on the panel using the sizing handles.

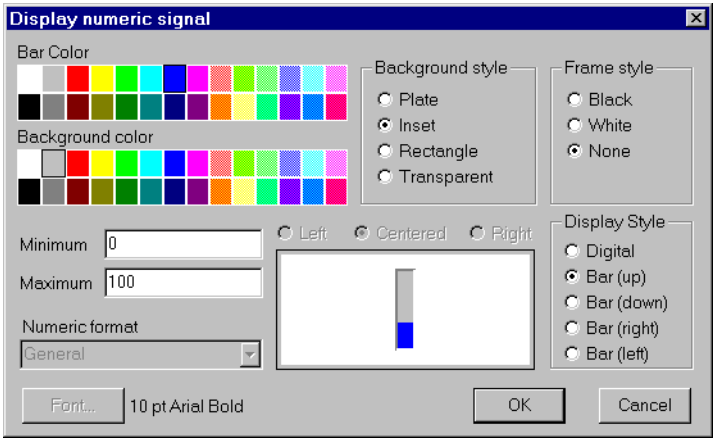


Now select **Insert»Expression…**.



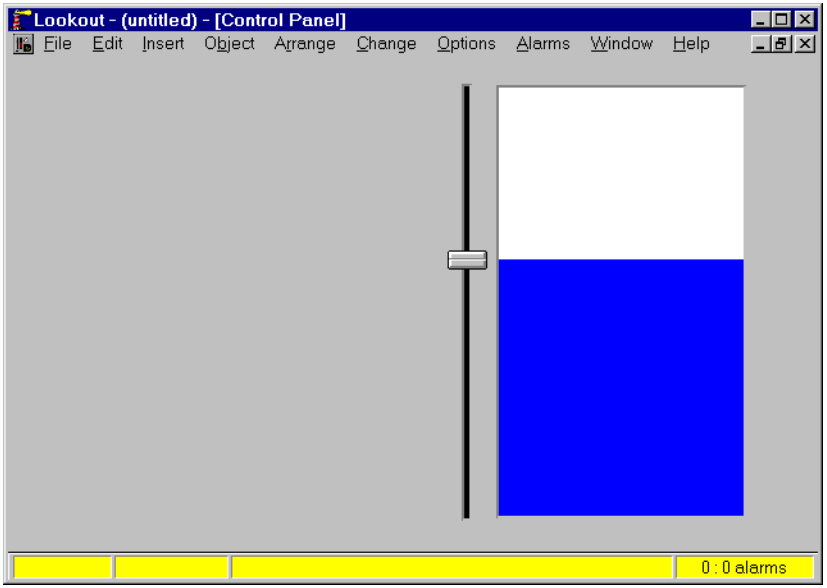
Pick `SimulationLevel` as the value to be inserted. Select the **Paste** button to insert the tagname in the expression field and select **OK**. You could also type the tagname in the expression field. Double-clicking on the tagname also pastes it to the expression field.

Lookout prompts you to select display characteristics for the numeric signal you just named. Pick the same parameters as below. The example uses dark blue as the **Bar Color** and gray as the **Background Color**. Notice the **Background Style** is **inset** and the **Maximum** is 100.



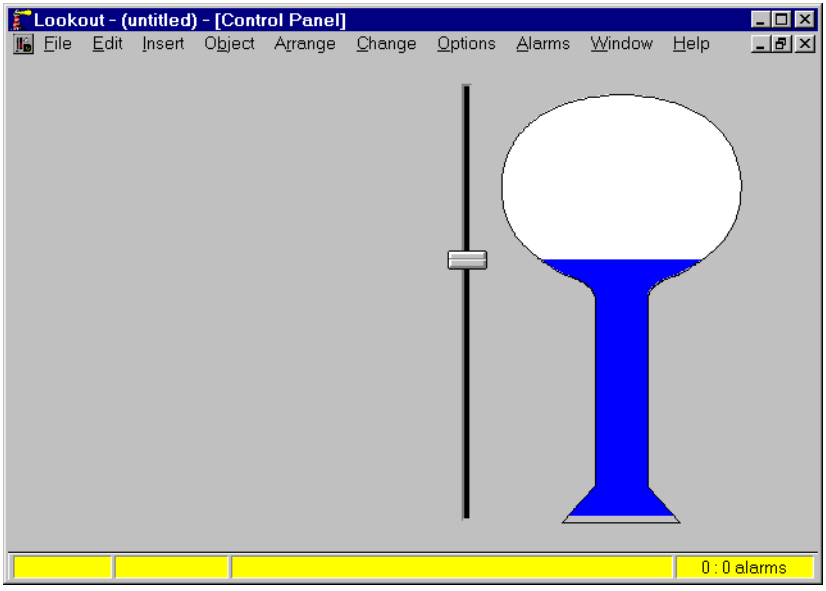
Note ***Lookout makes extensive use of color grids in many of its dialog boxes. These color grids may look slightly different from computer to computer.***

After selecting **OK** in the Display Numeric Signal dialog box, Lookout places the bar graph on the control panel. You can resize or move it around the screen. Arrange your control panel to look like the example.



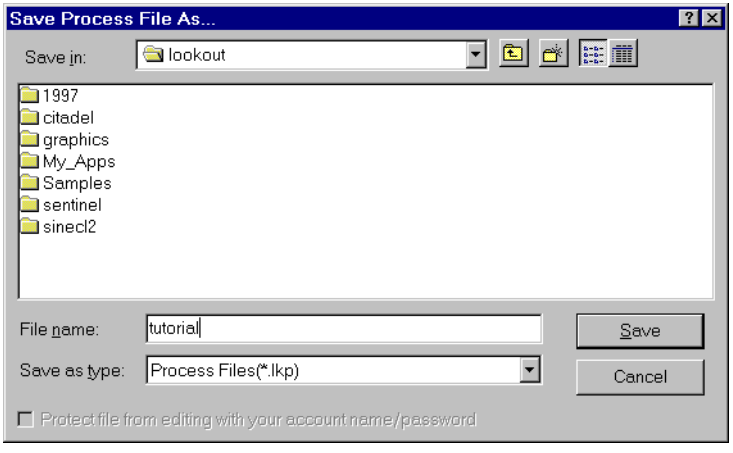
This bar graph simulates the water level in the tank. To give it a more realistic look, lay a graphic of a tank on top of it to act as a mask.

Select the **Insert»Graphic…** command and pick the graphic file `tank1_a.wmf`. Because the example uses a metafile you can resize it to just fit over the bar graph, masking out part of the bar graph.



Press <CTRL-SPACE> to toggle out of edit mode and move the slider up and down to see the fill effect created with your tank mask and bar graph. Then get back into edit mode so you can create a trend of the tank level.

Save the work you have already done before going any further. You should get into the habit of periodically saving your process file when developing a new system or making modifications to an existing one. Select **File»Save** from the menu commands, type `tutorial` for your filename, and select **OK**.

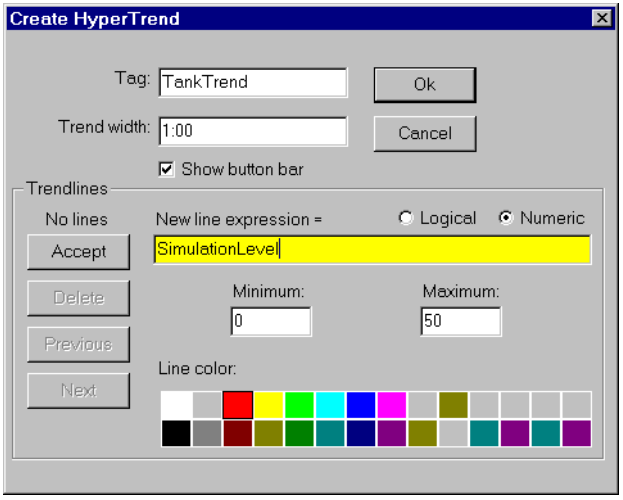


Your work is now stored on the hard drive as `TUTORIAL.LKP`. You should see the new filename in the Lookout title bar followed by the active control panel title.

Adding Data Display

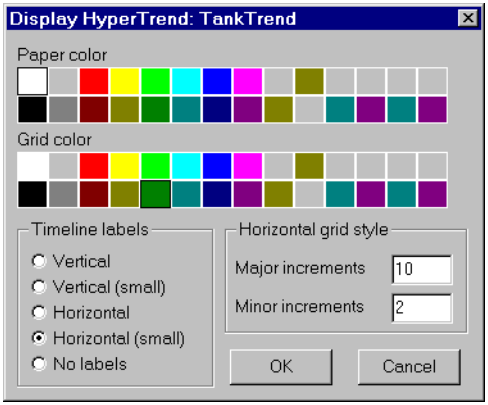
Select **Object»Create…** from the menu commands and pick the HyperTrend class.

Complete the parameters as shown. If your trend has a **Show button bar** option, select it.

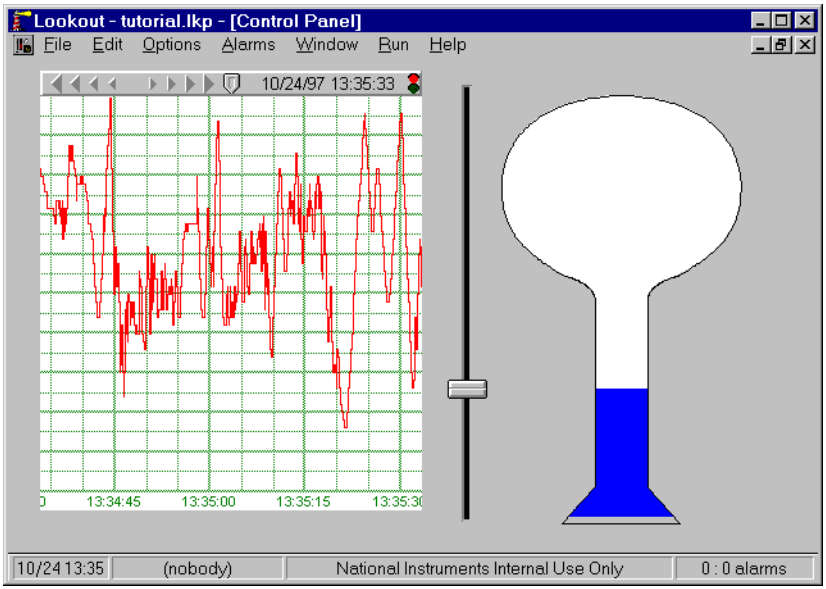


Notice that the examples defines **Trend Width** to be just 1 minute in length. This is so you can instantly observe fluctuations in your simulation pot. Normally the trend width would be much longer—1 day or 1 week for example.

After entering the value you want to trend (`SimulationLevel`), be sure to select the **Accept** button. Now select the **OK** button. The display dialog box appears.



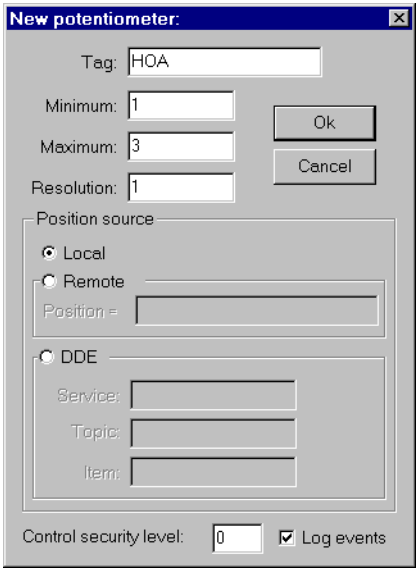
Choose your display parameters and select **OK** again. Like all other objects, you can resize and/or move the trend on your control panel.



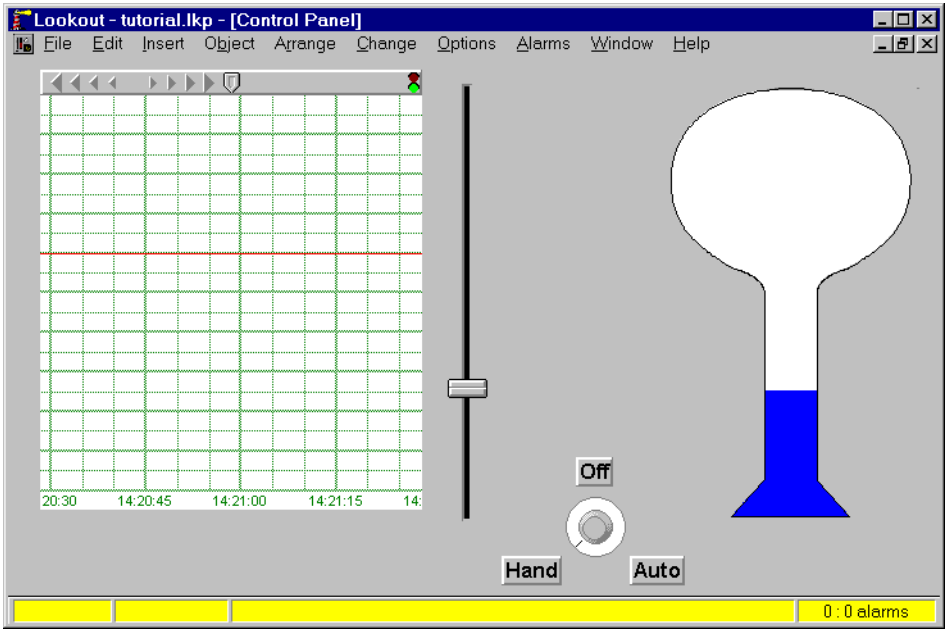
Again toggle out of Edit mode and move the slider up and down to see the effect. (You might also want to save your file.)

Adding Control Objects

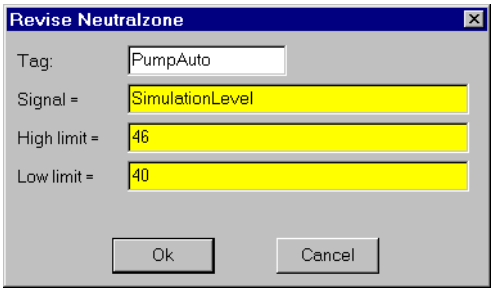
Now it is time to create an HOA switch and a Neutralzone object to control your pump. Get back into edit mode and create a Pot object. This serves as your three position switch (1=Hand, 2=Off, 3=Auto). Choose knob as the display style.



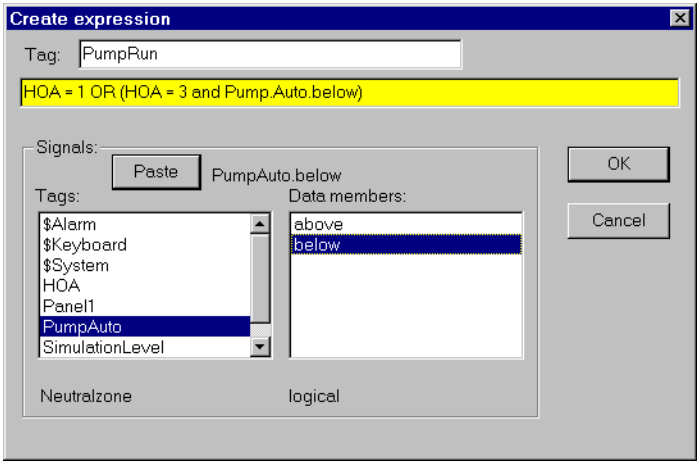
After inserting the pot, add appropriate text with the **Insert»Text/plate/inset…** command. Then lasso the text and knob and group them with the **Arrange»Group** command. You can now move the entire group of objects as a single item.



Now create a Neutralzone object. Call it `PumpAuto`.

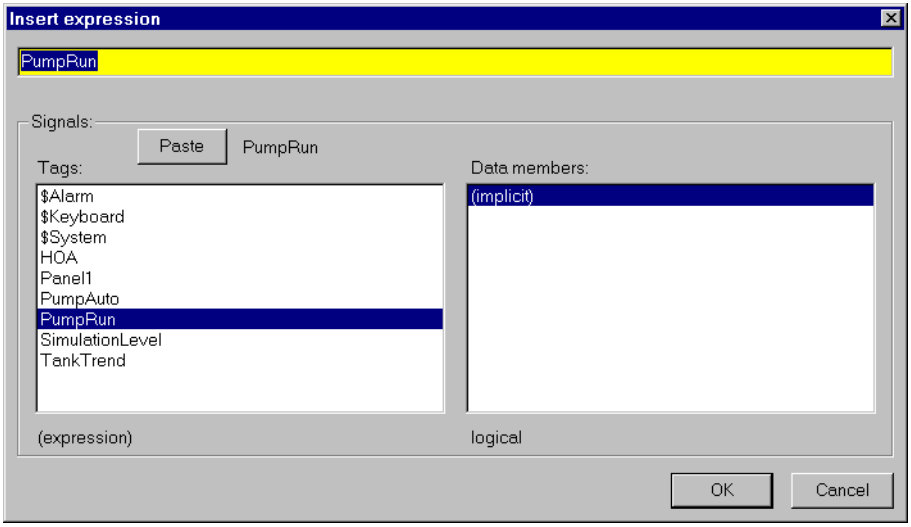


The next step is to create an Expression object. Call it `PumpRun`.

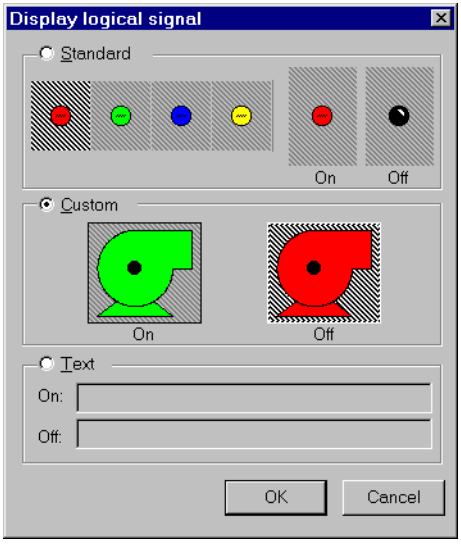


Notice the logic of the expression. If the HOA switch = 1 you are in the Hand position and want the pump to run, *OR* if the HOA switch = 3 you are in the Auto position *AND* you need to determine if the Neutralzone is calling for the pump to run (*below* specifies the signal is below the low limit and you need the pump to start). If the HOA switch = 2, you are in the Off position and the result of the expression is false.

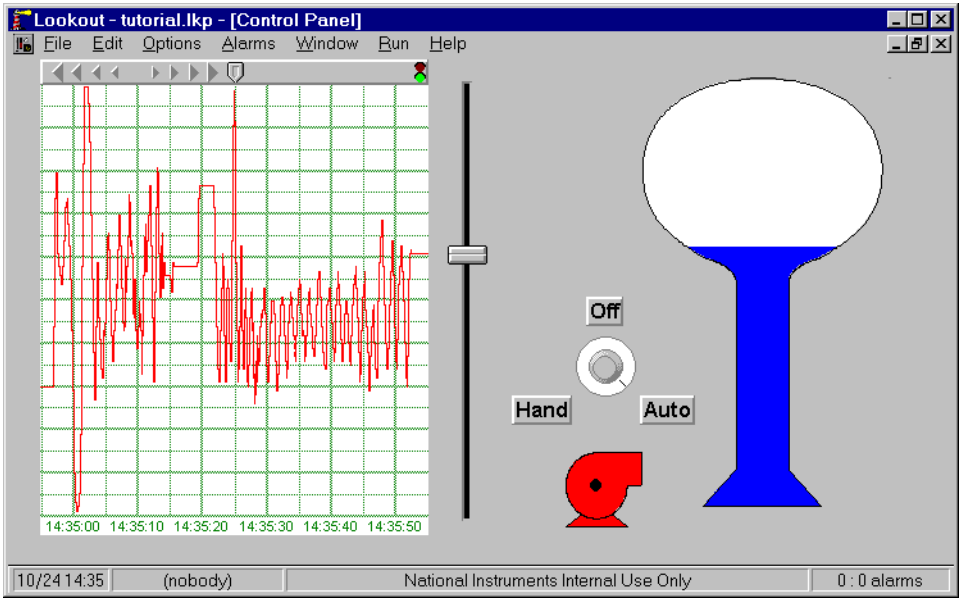
Select the **OK** button and Lookout presents the Insert Expression dialog box. Be sure to pick `PumpRun` if it is not already in the window and select **OK**.



The result of `PumpRun` is a logical signal. Because of this you can choose two graphic files to be displayed—one for the “on” signal and one for the “off” signal. Click on the **Custom** selection and scroll through the **On** list box until you find the `PMP1_RD.wmf` file. Then scroll through the **Off** list box and select `PMP1_GRN.WMF`. After you choose the appropriate files, select **OK**.



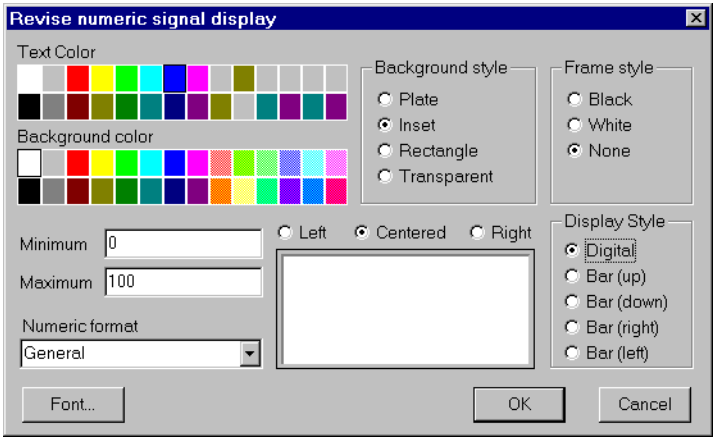
As before, the example uses two windows metafiles that can be resized by dragging a sizing handle. Resize and position the pump as shown in the following illustration.



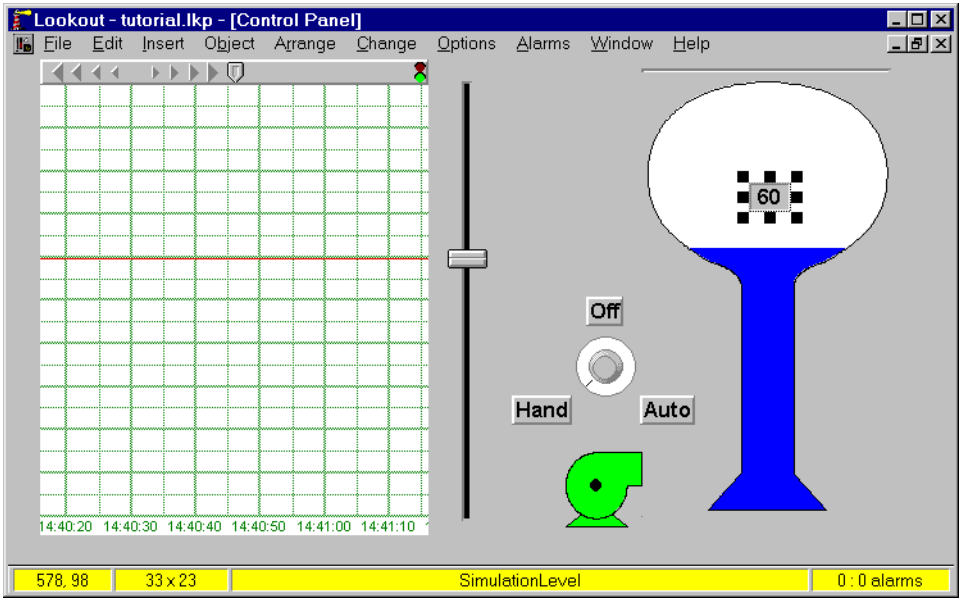
Before you leave Edit Mode and test your system, add one final item to see a digital reading of the water level in the tank. You could insert a new expression and choose digital as the display style; however, use a Lookout shortcut instead.

Click on the tank—the phrase `tanks\tank1-a.wmf` appears in the status bar. This tells you what you selected. Because you have stacked the graphic on top of the bar graph you must click one more time to select the next item down in the stack. The status bar should now read `SimulationLevel`. You can now quickly copy and modify the expression display parameters. Hold the <Shift> key down while dragging the bar graph (this type action is called shift-drag). This makes a copy of the bar graph. See *Mouse Shortcuts* in Chapter 3, *Getting Started*, for instructions on using the mouse. Now select and position the cursor over the new bar graph. Click the right mouse button to modify the display properties.

Complete the dialog box as follows and select **OK**.



Now resize the digital display and position it over the tank.

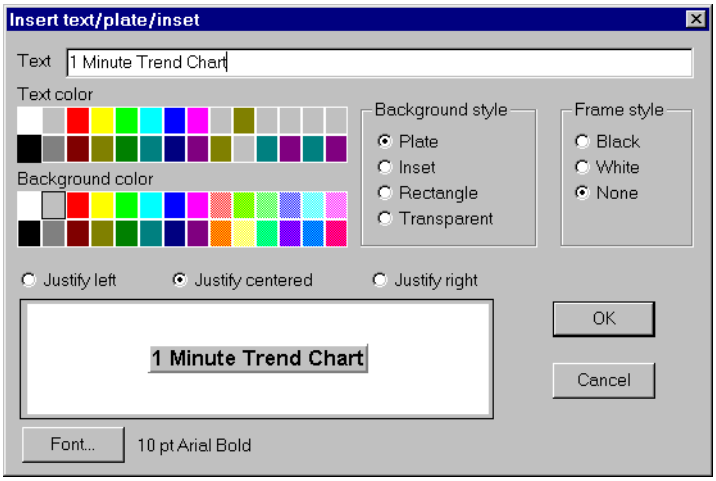


Toggle out of edit mode so you can test your system. Turn the HOA switch to the Auto position and run the slider up and down to see if the Neutralzone is working correctly. Then turn the HOA switch to the Off and Hand positions to complete your testing. The pump should change colors according to your logic.

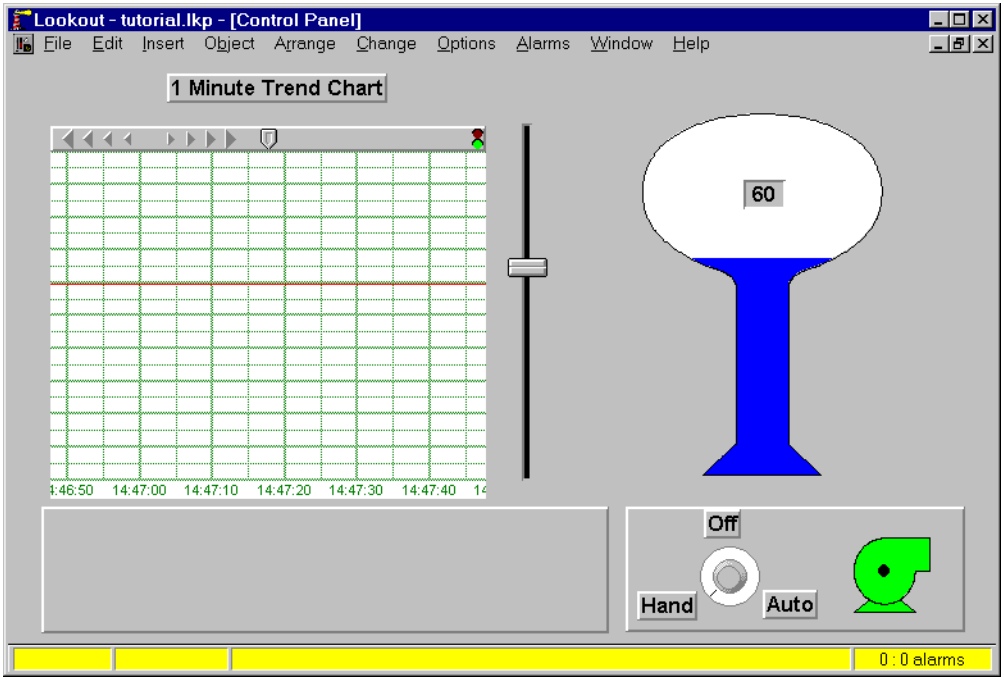
Completing the Interface Panel

Now return to edit mode and dress up the screen with plates, scales, text, and so on.

First select **Insert»Text/plate/inset…** and choose **Plate** as the background style. Do not enter any text in the window. Select **OK**. This is an extremely easy and useful technique for quickly creating different color plates and insets. Spend a little time experimenting with different combinations of plates, insets, and background colors.

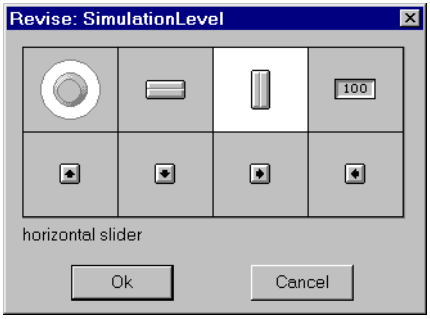


After inserting the plate, move it to the top left corner of the control panel and drag the sizing handle to stretch the plate. Then move the plate behind everything else with the **Arrange»Move to Back** command. You can then resume the positioning and sizing of the plate or other objects.



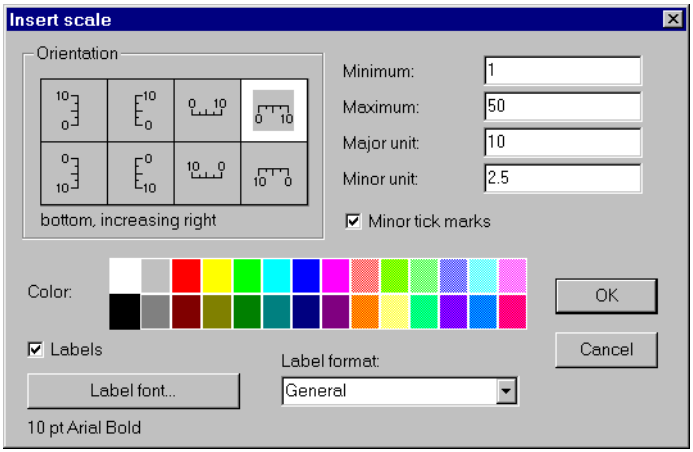
Continue adding plates, insets and text and in a few minutes your screen should look something like the example. You may need to reposition or resize several existing objects.

Change the vertical slider to a horizontal slider and reposition it. Select the slider with the left mouse button and then click the right mouse button to revise its display parameters. See *Mouse Shortcuts* in 3, *Getting Started*, for mouse operation instructions.

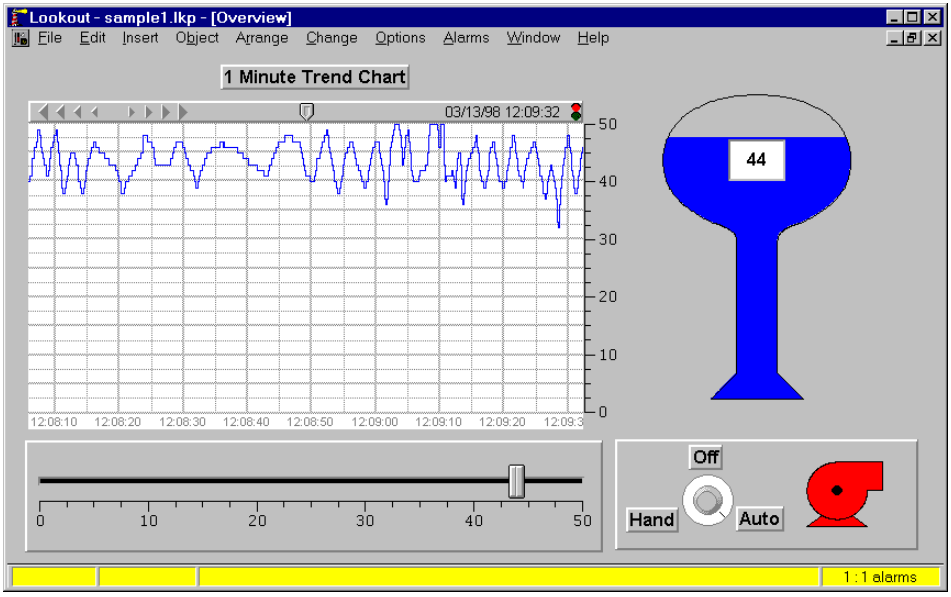


After moving the slider to the bottom of the control panel, add scales—one for the slider and one for the trend.

Select **Insert»Scale…** and complete the dialog box accordingly.



After positioning the scale next to the trend, shift-drag the scale to create a copy. Then click the right mouse button to modify the second scale for your horizontal slider.

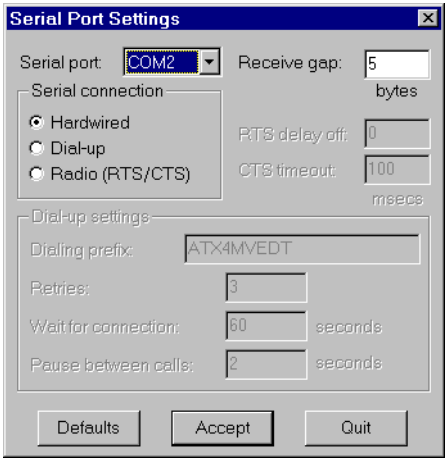


Again toggle out of Edit mode and test the system by moving the slider and knob.

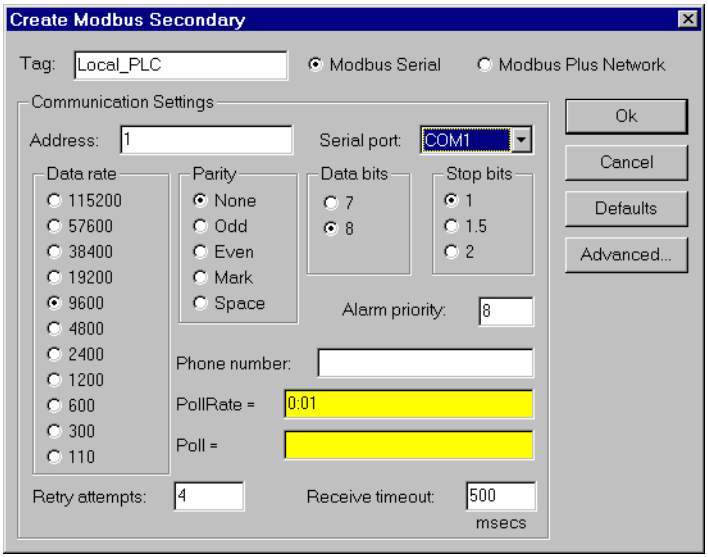
Of course you do not have to go through the preliminary steps of creating simulation signals on future projects. However, many times it is a good idea to test your logic before connecting the real thing.

Connect the PLC and RTU

Start the steps necessary to connect your PLC and RTU to Lookout. Begin by configuring your communication ports. Select the **Options»Serial Ports…** command from the menu. Configure COM1 for hardwired serial connection to your local PLC and select **Accept**. Then select COM2 from the **Serial Port** list box and pick **Radio (RTS/CTS)**. You use this port to talk to the remote RTU at the elevated tank. Select **Accept** again and you are finished—now select **Quit**.



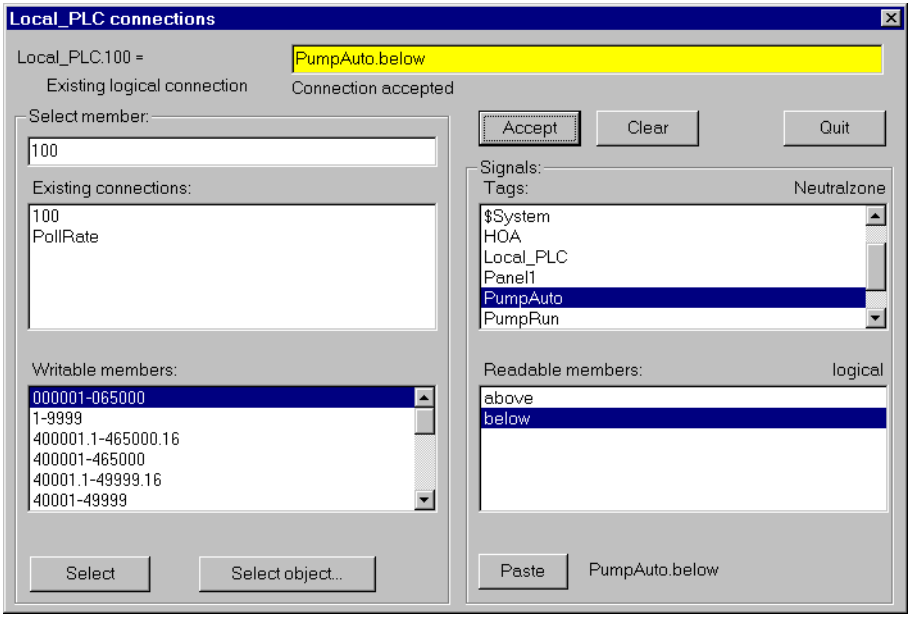
Now connect a Modbus protocol controller to your process. First you must create a Modbus object with **Object»Create…**.



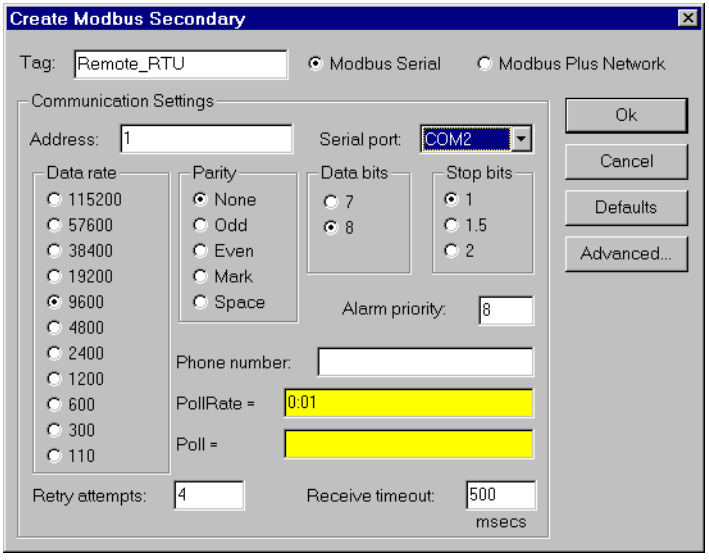
This is the local PLC located in the plant that controls the operation of the pump. Remember, you are hardwired to this PLC using COM1.

The next step is to connect the neutralzone signal to a coil on the PLC. This signal turns the pump on and off. Select **Object»Edit Connections…** from the menu and pick `Local_PLC`.

First, enter `100` in the **Select member** data field. Then click on the **Select** button. This tells Lookout that you are going to connect a signal to coil `100` on a Modbus device named `Local_PLC`. Now type in `PumpAuto.below` in the top expression window, and select the **Accept** button. You just connected an expression to coil 100 and a message appears confirming this.



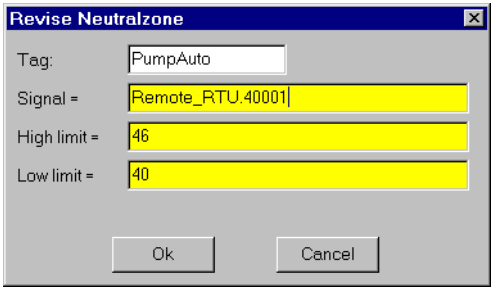
You now need to create another Modbus object for the remote RTU. Each controller has its own respective object in Lookout. Each of these objects determines its polling scheme, read/write blocking, and so on.



Because you are communicating through a remote radio to the RTU, you must select a serial port configured for radio communications. In this case, it's COM2.

All you have left to do is replace the signal in the neutralzone object from `SimulationLevel` (the slider) to the analog input coming from the RTU at the elevated tank. Select **Object»Modify…** from the menu and pick `PumpAuto`.

Enter `Remote_RTU.40001` to read the TankLevel coming from the controller as a 16-bit integer value.



You now have a full-blown working process file. Although you are performing the relatively simple task of turning a pump on and off based on a tank level, you learned many of the basic principles behind the object-oriented structure of Lookout. You also covered several shortcuts for copying and modifying graphics and expressions.

☞ **Note** ***It is important to remember that Lookout remains on-line—polling PLCs and RTUs, logging data, alarming, executing control logic, and so on—even when you are in edit mode. You never have to shut down your physical system to modify your Lookout application—this includes adding or deleting physical I/O, PLCs, RTUs, loop controllers, and so on.***

Conclusion

There are only a few basic principles to learn if you want to master Lookout—the most important being the concept of an object. Once you realize what objects are, how they behave, and how to use their data members, you should be able to quickly learn Lookout and begin building your application.

You do not need to memorize every object class in Lookout, but you should scan their definitions and data members to familiarize yourself with their functionality.

To review, an object bundles functionality with an integrated database. An object has parameters that define how it works. You define these parameters when you create or modify the object.

In the predefined object database, the data members can be numeric, logical, or textual. Data members can generate signals (that is, they are readable), receive signals (that is, they are writable), or both.

As a final analogy, think of the many object classes in Lookout as a comprehensive set of tools. Each object is designed to solve a particular problem. However, you must open the toolbox (documentation) to learn what is available, in order to become an efficient developer.

Part II

Lookout Features and Services

Chapters 6 through 17 describe the features and services of Lookout in detail. These chapters will familiarize you with the details of how to do many of special things you may need to do with Lookout.

- Chapter 6, *Serial Communications*, describes serial communications, namely COMSUB, and details how to define settings for three different serial connections: hardwired, radio (RTS/CTS), and dial-up.
- Chapter 7, *Expressions*, explains the features and uses of Lookout expressions, which are flexible, real-time math statements.
- Chapter 8, *Graphics*, describes adding static and dynamic graphics to a control panel and creating and using custom graphics.
- Chapter 9, *Alarms*, describes generated alarms and configuration services provided by the Lookout alarm subsystem. As a Lookout environment service, the alarm subsystem filters, displays, logs, and prints alarms.
- Chapter 10, *Security*, describes Lookout accounts and the environment service security subsystem, which oversees process file security, control security, viewing security, and action verification. With this system, you selectively determine which operators control particular objects, which operators view particular control panels, and which objects prompt the operator for command verification.
- Chapter 11, *Logging Data and Events*, describes three Lookout methods for logging real-time system data to disk—Spreadsheet Logger, Citadel Threaded Database Logger, and Event Logger—and report generation.
- Chapter 12, *Structured Query Language*, describes Structured Query Language (SQL), Open Database Connectivity (ODBC), and accessing Citadel data using both SQL and ODBC.

- Chapter 13, *Dynamic Data Exchange*, explains how to use Dynamic Data Exchange (DDE) with Lookout. DDE is the Microsoft message-based protocol used by applications like Microsoft Excel and Lookout to link to data in other applications.
- Chapter 14, *Networking*, explains how to use Lookout to monitor and control your process from any workstation (node) on the network.
- Chapter 15, *Redundancy*, describes how to configure two process control computers for redundancy, providing automatic transfer of control should the primary computer fail.
- Chapter 16, *Runtime Menu Commands*, describes Lookout menu bar pull-down commands available in Normal mode (that is, not Edit) mode.
- Chapter 17, *Edit Mode Menu Commands*, describes Lookout menu bar pull-down commands available in Edit mode. You use Edit mode to perform all system configuration and editing.

6

Serial Communications

This chapter describes serial communications, and details how to define settings for three different serial connections: hardwired, radio (RTS/CTS), and dial-up.

Introduction to Driver Objects

Certain object classes represent and communicate with external physical devices such as PLCs, RTUs, and controllers. A few examples include Modbus, Tiway, AB_PLC5, and Optomux. We use the generic term *driver* to refer to these types of object classes. The functionality built into driver objects enables them to communicate with the physical devices that they represent. Lookout communicates with the outside world primarily through driver objects.

In traditional systems, drivers are separate applications running independently of the operator interface. Driver programs compete for CPU time with applications such as database managers, MMIs, and historical data loggers, necessitating multitasking and increased CPU power. In contrast, Lookout drivers are not separate applications. Lookout driver objects work as any other object in the Lookout event-driven environment, except that they communicate with external devices.

With traditional systems, you assign a particular driver to a specific serial port. In such configurations, multiple drivers cannot share a single serial port. Lookout does not associate baud rate, data bits, parity, or stop bits with a particular serial port. In this way, drivers that implement different protocols and baud rates can use the same port and the same modem or radio frequency.

This capability allows you to mix and match RTUs, PLCs, and other devices over a single radio frequency without communication conflicts or special hardware. For example, you can use a single two-way radio connected to a serial port to communicate with several different brands of RTUs out in the field, each one using a different protocol. You can have seventy-five remote PLCs share a set of five dial-up modems.

All this is possible because of the Lookout *communication service*. Objects use the communications service, an environment service, to gain access to serial ports in an orderly and timely fashion.

☞ **Note** ***Some Lookout driver objects communicate with physical devices through dedicated hardware. These driver objects do not use serial ports but instead rely on their own proprietary network cards for interfacing to the outside world. A few examples include Modbus Plus (SA85 card), Data Highway (KT card), and DeltaTau (PMAC card). You do not need to configure serial ports for these objects classes. Refer to the appropriate object class documentation in Chapter 18,*** *Object Class Reference,* ***to verify if a particular object class uses a serial port.***

Understanding the Communications Service

The Lookout serial communication service, allocates serial port usage between driver objects. At the frequency of the object **Poll Rate,** a driver object notifies the communications service that it needs to use a specific serial port to poll a device. If the requested serial port is not in use, Lookout allocates the serial port to the driver object. When the driver object takes control of the serial port, it defines port communication parameters such as baud rate and protocol and polls its device. When polling is complete, the driver object releases the port so the communications service can allocate it to other driver objects.

You can uniquely configure each serial port for hardwired, radio, or dial-up communications through the **Serial Port Settings** dialog box. Refer to the *Defining Serial Port Settings* section for more detailed information.

☞ **Note** ***You must define serial port communication settings on every type of Lookout product, including Development/Runtime, Runtime-Only, and Express systems.***

Defining Serial Port Settings

This section walks you through the steps to configure serial port settings for hardwired, radio, and dial-up communications.

1. From the Lookout menu bar, select **Options»Serial Ports…** The **Serial Port Settings** dialog box (Figure 6-1) appears.

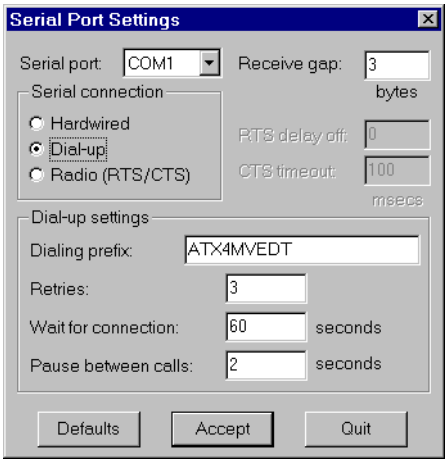


Figure 6-1. Serial Port Settings Dialog Box

2. In the **Serial port** data field, select the communication port you are defining. In this example, `COM1`.
3. Define the serial port parameters for the appropriate communication port. Refer to the remaining sections in this chapter for complete descriptions of the parameters.
4. Click on **Accept** to save the parameter changes for the serial port.
5. Click on **Quit** to exit the dialog box.

Selecting the Serial Port

The **Serial port** data field is a drop-down list box. Use it to select the communication port you are defining. Microsoft Windows supports up to nine serial ports; however, most computers support only two serial ports without additional hardware.

Setting Receive Gap

The **Receive gap** setting is available for all serial connection types. This number specifies the number of empty bytes (or amount of time) a driver receives from a controller before the driver recognizes the end of a message frame and asks for another message. Normally you should leave this at the default setting of 5. However, if you are experiencing garbled communication alarms, you might try increasing this number to allow more dead time before Lookout decides it has received a complete message. For example, with a slow baud rate of 1200, you might have to increase the **Receive gap** setting to approximately 30.

Selecting the Serial Connection

Hardwired Settings

Hardwired serial connections require no hardware handshaking for line control. Use this setting for all serial communication types except dial-up telephone and remote radio transceivers. You should also use this setting when directly connecting Lookout to the Master Repeater on a radio system or through a leased-line modem. Because a Master Repeater is a full duplex device that does not require keying and unkeying of the frequency, it acts much like a physically hardwired network. Other hardwired connection types include RS-232, RS-422, RS-485, and leased telephone lines.

RTS/CTS Handshaking Settings

RTS/CTS is a local hardware handshaking mechanism between the local computer and the local communication device. Use the **Radio (RTS/CTS)** serial connection when connecting the serial port to a device that requires RTS/CTS hardware handshaking, such as a radio transceiver that must be keyed up during data transmission and unkeyed during data reception. Other half-duplex communication media such as RS-485 may require RTS/CTS hardware handshaking. Although the RTS/CTS scheme works identically for other RTS/CTS communication schemes, assume that you are communicating via radio for this discussion.

When you select RTS/CTS hardware handshaking, Lookout controls the RTS, or request-to-send pin, and monitors the CTS, or clear-to-send pin, during data transmission (pins 4 and 5 on a 25-pin RS-232 connector). Therefore, you must have at least the RTS pin (pin 4) wired *straight through* on your RS-232 cable. The CTS pin (pin 5) is optional.

Lookout initiates a serial transmission on an RTS/CTS port by first asserting RTS to key the radio. Lookout then begins monitoring the state of the CTS pin. When the radio transmitter is fully keyed and ready to transmit, the radio asserts CTS and Lookout immediately begins data transmission. If the radio does not assert CTS within the **CTS timeout** setting (default is 100 msecs), Lookout assumes the radio is ready to transmit and transmits anyway.

The **CTS timeout** setting is the maximum amount of time that Lookout waits after asserting RTS for CTS before transmitting. Most radios typically take between 10 and 80 milliseconds to key up. Consult your radio specifications and DIP switch settings to determine the key-up delay on your radio.

If your radio can assert CTS when it is ready to transmit, add about 50 milliseconds to the radio key-up delay specification and use this total value for the **CTS timeout**. If your radio does not assert CTS, you should begin by adding about 20 milliseconds to your radio keyup time. Then increase this value in 10 millisecond increments until the remote radio begins to correctly receive the first bytes of the message.

Some radios may assert CTS before they are actually ready to transmit. In this case, disconnect the CTS line (pin 5 on a 25-pin RS232 connector) and set the **CTS timeout** to a value high enough to let the radio fully key before transmission.

After it transmits the last byte of data, Lookout continues to assert RTS, keeping the radio keyed until the **RTS delay off** time period expires. You should set this value to the default of zero milliseconds so that Lookout unkeys the radio as soon as possible to prepare to receive the response.

When unkeyed, most radios generate an audible squelch tail that the remote device might decode as unexpected garbage bytes. Some remote devices reject the entire message instead of just decoding the valid data and ignoring the extra garbage bytes. In this case, keep the radio keyed for several milliseconds using the **RTS delay off** setting. This time period delays the squelch tail long enough for the remote device to recognize the last data frame as valid before receiving garbage bytes caused by the squelch tail.

If you set the **RTS delay off** setting too high, the remote device begins transmitting its response before the local radio is unkeyed, causing a communication alarm in Lookout.

Dial-Up Modem Settings

Use the **Dial-up** serial connection when you use a modem in conjunction with a switched telephone line (not leased line). You can customize the dial-up settings for your particular modem and phone line.

The default **Dialing prefix** settings are based on the Hayes Corporation AT command set, which is an industry standard for data modems. The following table explains the Lookout default settings. For additional commands, refer to your modem operation manual.

Table 6-1. Dialing Prefix

AT	Attention code that must precede all commands
D	Dial phone number with these modifiers: P for pulse; T for tone
En	Local echo mode: E for no echo
Mn	Speaker on or off: M for speaker always off
Vn	Verbal or numeric result codes: V for numeric result codes
Xn	Result code and dialing options: X4 waits for dial tone before dialing, and recognizes busy signal

When you use an external dial-up modem with Lookout, the DTR line in your cable between the modem and the computer must be wired straight through. This line is pin 20 on a 25-pin RS-232 connector and pin 4 on a 9-pin connector. Lookout uses the DTR line to command the modem to disconnect (hang up) and return to the command mode.

Some factory modems are not configured to respond to the DTR line. After Lookout first successfully dials out to a remote modem and finishes the polling cycle, it drops the DTR line but the modem remains connected. If the modem does not respond after several seconds of Lookout attempting to raise and drop the DTR line, Lookout generates an alarm stating that the modem is not responding. If you receive this alarm message, your modem is not configured to monitor the DTR line.

The Hayes Corporation standard command for configuring the modem to hang up and enter command mode upon loss of DTR is &D2. You can use a terminal program to make this setting permanent on most modems by entering the modem command AT&D2&W to store the setting permanently in nonvolatile modem memory. Or you can just add &D2 into the **Dialing prefix**. The default **Dialing prefix** is ATX4MVEDT, so you might change it to AT&D2X4MVEDT.

Retries specifies the number of times Lookout dials the specified phone number and attempts to connect to the modem at the other end of the line. If Lookout fails to connect after the specified **Retries**, it generates an alarm and moves on to the next phone number in the polling queue (if a queue has formed).

Wait for connection specifies the length of time Lookout waits to receive a connect signal back from the modem it is calling. The time period begins when Lookout first sends the local modem the dialing prefix command. The time should be long enough for the local modem to receive a dial tone, dial the phone number, allow the remote modem to pick up the line, and send back a connect message. If the specified time is too short, your system could be operating correctly but never make a connection.

Pause between calls is the length of time Lookout waits after hanging up before it sends the local modem the next dialing prefix signal. If the specified time is too brief, your system may not hang up the existing call but still instead attempt to call the next number.

☞ **Note** ***Your specific modems, radios, and local phone lines may operate faster or slower than the default settings. You may need to use a trial-and-error approach to find the best settings for your system.***

7

Expressions

This chapter explains the features and uses of Lookout expressions, which are flexible, real-time math statements.

To work effectively in Lookout, you must understand Lookout expressions. They resemble spreadsheet-style formulas with a mixture of constants and variable signals from objects. They can be short and simple, or extremely complicated with several signal inputs, function calls, and multiple levels of parentheses. Lookout provides an extensive library of built-in functions that includes logical, mathematical, statistical, text, trigonometric, and time functions.

A single expression may incorporate any number of numeric, logical, and text signals within its calculation. However, the *result* of the expression can be only one of three types: numeric, logical, or text. If you attempt to create an illegal expression, such as adding a logical value to a numeric value, Lookout beeps and displays an error message. The outermost function or operator in the expression returns a variable type that determines the overall signal type of the expression.

Simple expressions consist of single value. Simple expressions do not contain any modification, manipulation, math, or logic (for example, a single object tagname like `Pot1`). When you add any functionality to a simple expression, you create a complex expression (for example, `Pot1` > 33).

The following examples are typical expressions. Depending on your system requirements, your expressions might be much more involved.

True

Always returns the value true (on).

Pot1

Returns the current value of Pot1.

879.03

Always returns the value 879.03.

1:04:33

Returns the value 0.0448264. If formatted in hours, minutes, and seconds, it is displayed as 1:04:33.

Pressure * 2.31

Multiplies Pressure by 2.31.

(Switch1 or Switch2) and TankLevel > 65.2

If Switch1 or Switch2 is true, and TankLevel is greater than 65.2, return true. Otherwise, return false.

Pressure >= Setpoint and !ReliefValve and TimeOccuring > 0:30

If Pressure is greater than or equal to Setpoint, and ReliefValve is false, and TimeOccuring is greater than 30 seconds, return true. Otherwise, return false.

nif(PLC1.AutoPos, AutoTemp, nif(PLC1.ManPos, ManualTemp,0))

If the alias member AutoPos of PLC1 is true, return the value of AutoTemp. If the alias member ManPos of PLC1 is true, return the value of ManualTemp. Otherwise, return the value 0.

tif(HOA=1,“Hand”,tif(HOA=2,“Off”,“Auto”))

If the HOA switch is at position 1, return `Hand`. If the HOA switch is at position 2, return `Off`. Otherwise, return `Auto`.

You can accomplish much the same thing with the following `tchoose` expression as you can with the previous `tif` expression

tchoose(HOA,"hand","Off","Auto")

If the HOA switch is at position 1, return `Hand`. If the HOA switch is at position 2, return `Off`. If the HOA switch is at position 3, return `Auto`.

Make sure that your expressions do not attempt to calculate illegal operations, such as dividing by zero or finding the arc cosine of a number greater than 1. If a signal can assume a value causing Lookout to attempt an illegal mathematical operation, you should specifically test for that condition in the expression. Expressions trap illegal mathematical operations and generate alarms in the Math alarm group. The alarms do not reset until you correct the expression. The following list includes some of the illegal conditions that Lookout traps:

- attempting to divide by zero
- taking the square root of a negative number

- numeric underflow
- numeric overflow

☞ **Note** ***If any value referenced in an expression changes when an even occurs, the expression automatically recalculates. This is the same event-driven concept that objects implement.***

Creating Expressions

Because Lookout uses expressions in many places, you can create expressions as independent objects, parameters in objects, connections to object data members, or you can insert expressions on control panels.

Through dialog boxes, you create expressions during application development. There are two types of data entry fields in Lookout dialog boxes—white and yellow. White data entry fields accept only constant values, and yellow data entry fields accept expressions.

☞ **Note** ***In dialog boxes, all yellow data entry fields accept expressions.***

Expressions on Control Panels

With the **Insert»Expression…** command, you graphically display the result of an expression on a control panel. However, this method does not create an object; therefore, it does not create an output signal that other expressions or objects can use. (There is no tag associated with the expression.)

With the shift-drag and shift-right-click mouse actions, you can modify expressions on control panels and their graphical display. See *Configuration Shortcuts* in Chapter 3, *Getting Started*, for specific instructions.

Every result type (logical, numeric, or text) has a corresponding display dialog box, as shown in Figure 7-1, Figure 7-2, and Figure 7-3. See Chapter 8, *Graphics*, for more information on displaying graphics.

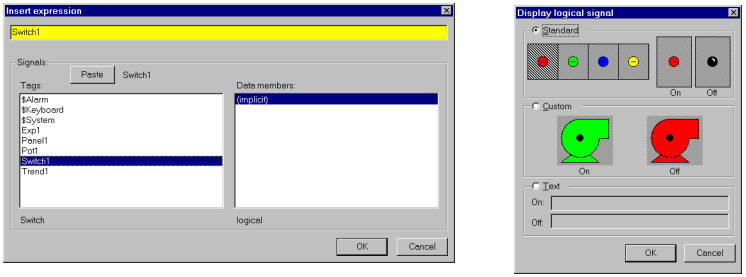


Figure 7-1. Example of **logical** Expression and Corresponding Display Dialog Box

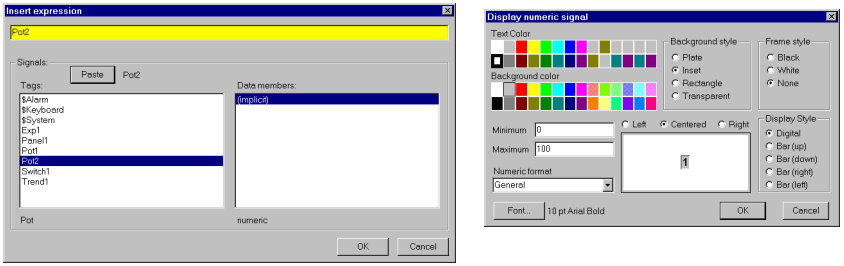


Figure 7-2. Example of **numeric** Expression and Corresponding Display Dialog Box

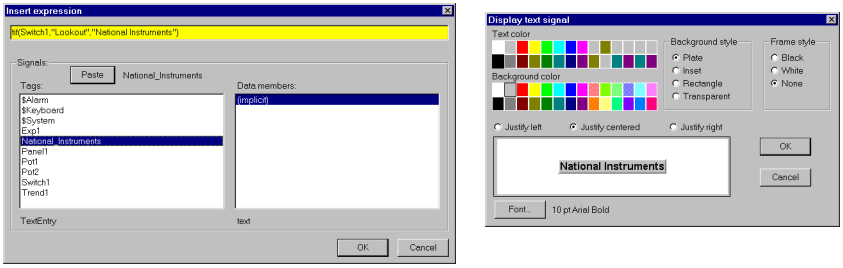


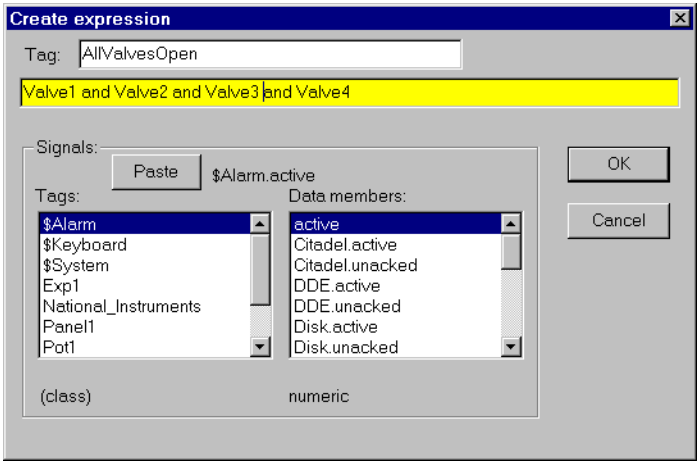
Figure 7-3. Example of **text** Expression and Corresponding Display Dialog Box

Expression Objects

With the **Object»Create…** menu command, you can create an Expression object. Like other object classes, the global (expression) object class requires a unique tagname.

Other expressions and/or objects can reference the output of an (expression) object. When you need to define a unique condition that your application uses multiple times, use an (expression) object. Instead of defining the same expression in many places, you can create it one time and use its tagname wherever the condition applies. For example, the

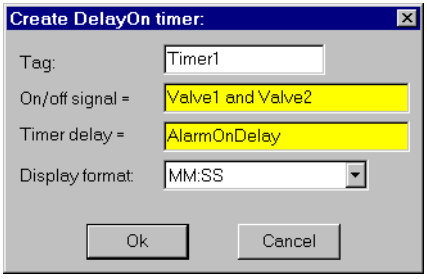
expression tag `AllValvesOpen` in the following **Create expression** dialog box is easier to reference than the long expression.



Expressions as Parameters

Many object classes accept expressions as parameters. When they do, the parameter expects the expression to return a certain signal type—numeric, logical, or text. Refer to the specific object class definition in Chapter 18, *Object Class Reference*, to verify the type of each parameter.

With the following dialog box, we can create a DelayOn object.



The parameter **On/off signal** expects a logical expression and **Timer delay** requires a numeric expression. Both `Valve1` and `Valve2` are tagnames for switches. Connecting them with *and* produces a logical result and satisfies the type condition of the first parameter. `AlarmOnDelay` is the tagname of a potentiometer used to adjust the timer delay setpoint, which creates a numeric signal, satisfying the type condition of the second parameter.

Because the **Timer delay** parameter is an expression, we have many configuration possibilities, including the following:

- Enter a constant. In this case, the delay never changes.
- Enter the name of an output signal from another object such as a Pot. In this case, the operator adjusts.
- Enter a complex expression that automatically calculates the delay based on multiple inputs.

Similar configuration solutions exist for the **On/off signal** parameter.

When an expression parameter fields is yellow, you can get help building your expressions. For example, assume that you cannot remember whether the valve tagname in the previous example was `Valve1`, `Valve_1`, or `ValveOne`. If you position the cursor over the yellow expression field and right-click, the **Expression Editor** dialog box appears. Using this dialog box, you can select the proper tagname (`Valve1`) and paste it directly into the expression field. Click on **OK** and Lookout writes the expression into the targeted parameter field.

Expressions as Connections

You can connect object data members with expressions. Writable data members accept expressions as inputs, much like parameters. To connect an expression to a data member, use the **Object»Edit Connections…** menu command. See Chapter 5, *Developer Tour*, for detailed information on connecting expressions to data members.

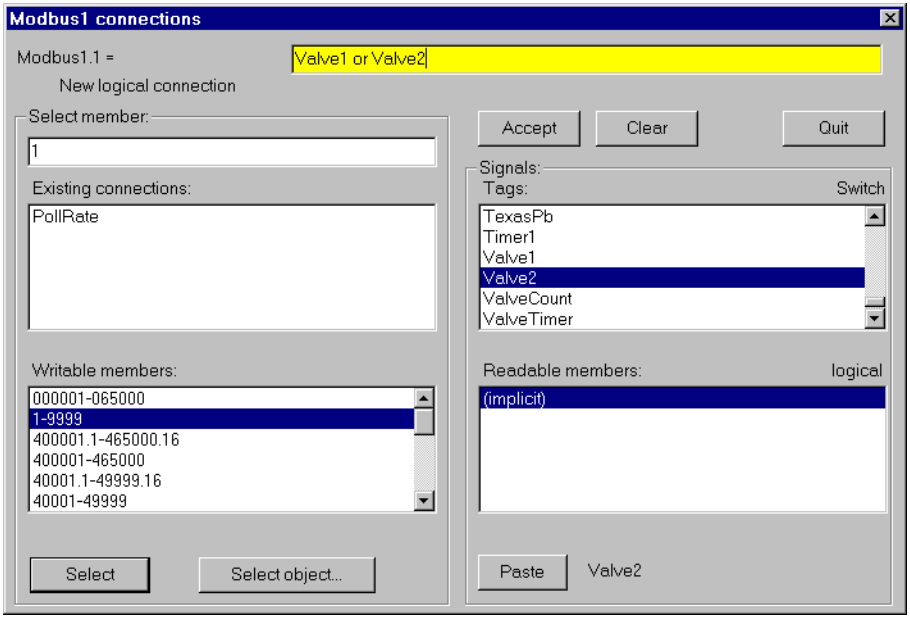


Figure 7-4. Edit Connections Dialog Box

Expression Dialog Box

When you right-click on a yellow parameter field, select the **Insert»Expression….** menu command, <Shift> right-click on an existing expression that appears as a graphic on a control panel, the expression dialog box appears.

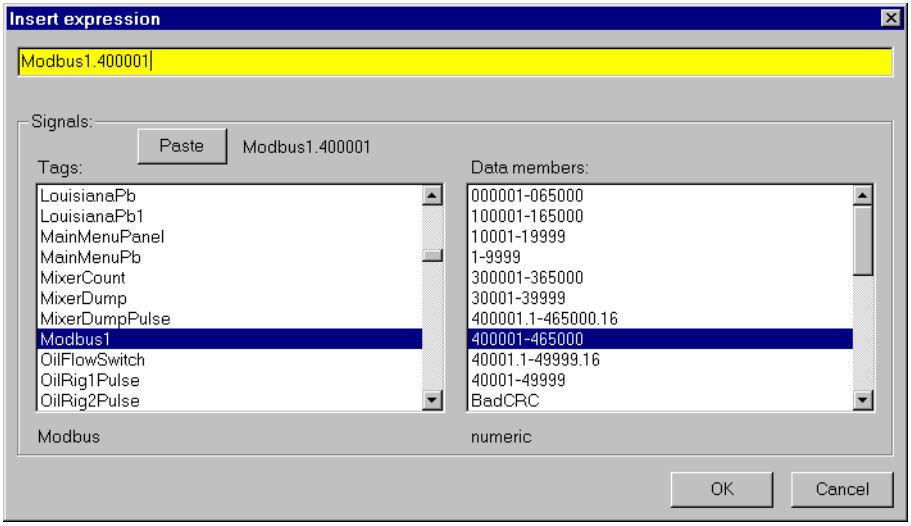


Figure 7-5. Insert Expression Dialog Box

There are three fields in the **Insert expression** dialog box in Figure 7-5. You can enter your expression in the top yellow field. The lower left field lists all object **Tags** that generate readable signals. The lower right field is a list of the readable **Data members** for the currently selected object. Notice that the dialog box indicates the *object class* selected in the **Tags** list (Modbus) and the *signal type* selected in the **Data members** list (numeric).

You can enter an expression directly or use the **Paste** button to select and insert tagnames in the expression field. As you select different tags from the list box, the name to the right of the **Paste** button changes accordingly.

Some objects have multiple readable data members, such as a Modbus object. Lookout concatenates the tagname, followed by a period and the selected data member.

Once you combine the desired tagname and data member, click on the **Paste** button. Lookout pastes the tagname into the expression field. Clicking on the **Paste** button a second time copies another instance of the tagname to the expression window—this time at the cursor.

☞ **Note** ***Instead of selecting a tagname and then clicking on the* Paste *button, double-click on the tagname. Lookout instantly pastes the tagname to the expression window for you. Many other dialog boxes use the same methodology.***

Obviously, you can manually type or modify a tagname, a data member, or a mathematical function in the expression field at any time. Because you might have many hard-to-remember, defined objects, the **Tags** list box serves as a quick reference for all of your previously defined objects.

Expression Syntax

In an expression, operators are instructions to perform an operation on a value or to combine values to form a new value. For example, a simple operator is the `/` symbol. It divides one value by another; the formula (`5 / 2`) reads *five divided by two* and produces a result of 2.5. The five and two in the preceding example are operands (the `/` operator requires numeric operands).

White Space

You can use tabs and spaces between operators, functions, and function parameters in Lookout to make expressions easier to read. In this manual, we use spaces between operators for added clarity.

We call spaces and tabs *white space* characters because they provide space between parameters. We maintain this practice for the same reasons that we use space between words and paragraphs in a book—to achieve greater organization and clarity. Because Lookout ignores white space characters, you can use white space characters to separate object names in an expression. However, you cannot embed white space characters within tags or alias names.

Arithmetic Operators

The following operators perform basic arithmetic operations. They require numeric operands and produce numeric results.

Table 7-1. Arithmetic Operators

+	Addition
–	Subtraction
*	Multiplication
/	Division
%	Percentage (divides preceding value by 100)
^	Exponentiation
–	Negation (additive inverse of the following value)

Text Operator

The text operator is the ampersand character (&). An ampersand joins two text strings. For instance, the expression `“Call me“ & “Ishmael”` produces a text signal of `Call me Ishmael`. Typical uses for the text operator include

- imbedding a numeric value within a text string
- using it in an action verification expression
- using it in an alarm message
- sending out to a remote PLC display panel

The expression `“Flow rate is” & TEXT(Flow, “0.00”)& “gpm”` produces a text signal of `Flow rate is 141.23 gpm`, assuming `Flow` is a numeric signal whose value rounds to 141.23.

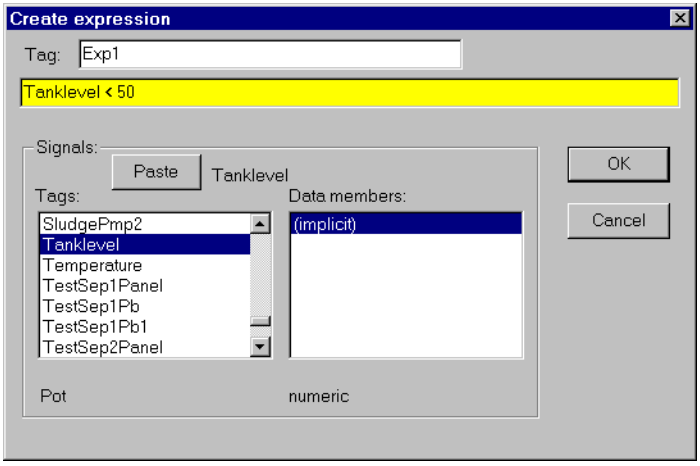
Comparison Operators

Comparison operators compare two numeric values and produce a logical result of either TRUE or FALSE (on or off).

Table 7-2. Comparison Operators

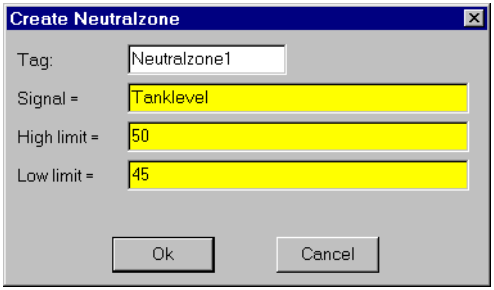
<	Is less than
>	Is greater than
<=	Is less than or equal to
>=	Is greater than or equal to
=	Is equal to
<>	Is not equal to

When setting up process control strategies, you often want to compare two numeric values. You can use the result of the following expression to control a tank fill valve: `TankLevel<50` is TRUE while `TankLevel` is less than 50 and FALSE while `TankLevel` is greater than or equal to 50.

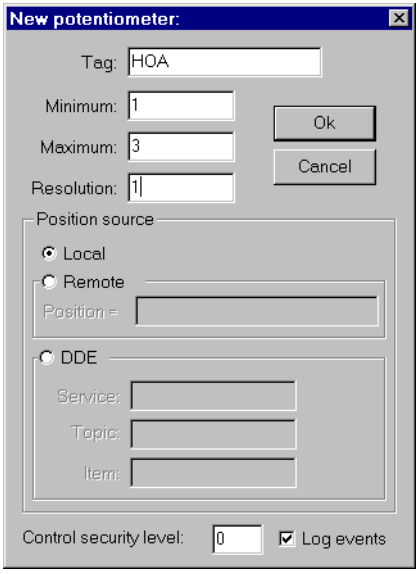


You can then connect this logical result to a PLC or RTU to control the fill valve. The valve opens when the tank level drops below 50 and closes when the tank level rises above 50. Notice, however, that this method can cause the valve to fluctuate between open and closed too often if the tank level hovers around 50.

Neutralzone is an object class with built-in dead band and is more appropriate for controlling the tank fill valve. The following example is a better usage of the comparison operator:

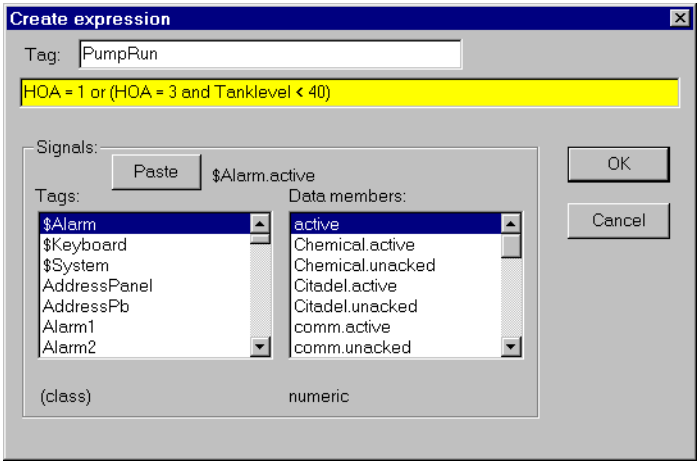


Avoid using the *is equal to*(=) and *is not equal to* (<>) comparison operators to compare an analog value from a PLC or the result of mathematical calculations in an expression. Because numeric (floating point) values have about 17 significant digits, `TankLevel = 40` might never be exactly true. If you know the signals are integer values, such as the numeric signal from the potentiometer in the following example, use the = and <> operators.



Notice from the following **Create expression** dialog box we are creating a three position switch with the Pot object class, where 1=Hand, 2=Off, and 3=Auto.

Now we can create an expression object that turns the pump on in two conditions: if the `HOA` switch is in the Hand position or if the switch is in the Auto position and the `TankLevel` is less than 40.



Notice that the HOA signal is exactly 1.00, 2.00, or 3.00.

Expression Functions

There are over fifty built-in expression functions, generally classified as follows:

- logical functions
- lookup functions
- mathematical functions
- statistical functions
- text functions
- trigonometric functions
- date/time functions

The remainder of this chapter describes each function, specifies the syntax, and provides an example on how to use each.

Logical Functions

AND	Syntax	*logical1* AND *logical2* AND *logical3*...
	Example	Switch1 AND Pot1 > 50.0
	Description	If Switch1 is on *and* Pot1 is greater than 50.0, return TRUE, otherwise return FALSE. Returns TRUE if *all* logical parameters are TRUE.
FALSE	Syntax	FALSE
	Example	False
	Description	Returns the logical value FALSE. Accepts no parameters.
LIF	Syntax	LIF (*logical*, *logical result1*, *logical result2*)
	Example	LIF(Switch1, False, Pot1 > 50.0)
	Description	If *logical* is TRUE, return *logical result1*, otherwise return *logical result2*. In the example, if Switch1 is on, it outputs the logical value FALSE. Otherwise, it returns the logical result of Pot1 > 50.0.
NIF	Syntax	NIF (*logical*, *numeric result1*, *numeric result2*)
	Example	NIF(Switch1, 99.9, Pot1)
	Description	If *logical* is TRUE, return *numeric result1*, otherwise return *numeric result2*. In the example, if Switch1 is on, it outputs the numeric value 99.9. Otherwise it outputs the value of Pot1.
NOT	Syntax	NOT (*logical*) or *!logical*
	Examples	NOT(*Switch1*) or *!Switch1*
	Description	If *logical* is TRUE, return FALSE, else return TRUE. In the example, if Switch1 is on, return FALSE, but if Switch1 is off, return TRUE.
OR	Syntax	*logical1* OR *logical2* OR *logical3*...
	Example	Switch1 OR Pot1 > 50.0
	Description	Returns TRUE if at least one logical parameter is TRUE. In the example, if either Switch1 is on or Pot1 is greater than 50, or both, the expression returns TRUE. If neither condition is true, it returns FALSE.

TIF	Syntax	`TIF (logical, text result1, text result2)`
	Example	`TIF(Switch1, "Text Message", TextInput1)`
	Description	If *logical* is TRUE, return *text result1*, otherwise return *text result2*. Notice the use of quotation marks. In the example, if Switch1 is on, it outputs `Text Message`; otherwise, it outputs the current text value of TextInput1.
TRUE	Syntax	`TRUE`
	Example	`TRUE`
	Description	Returns the logical value TRUE. Accepts no parameters.
XOR	Syntax	*`logical1`* `XOR` *`logical2`* `XOR` *`logical3...`*
	Example	`Switch1 XOR Pot1 > 50.0`
	Description	Returns TRUE *if only one* logical parameter is TRUE. In the example, If Pot1 is greater than 50 and Switch1 is on, it outputs FALSE because both logical parameters are TRUE. If Pot1 is less than 50 and Switch1 is on, it outputs TRUE because only one logical parameter is TRUE.

Lookup Functions

LCHOOSE	Syntax	`LCHOOSE (`*`numeric`*`, `*`logical1`*`, `*`logical2`*`, `*`logical3`*`,...)`
	Example	`LCHOOSE(Pot1, Switch1, TRUE, Pb1)`
	Description	Returns the `logical` parameter corresponding to the integer portion of the `numeric` parameter. If the `numeric` parameter is less than 2.0, `LCHOOSE` returns `logical1`. If the `numeric` parameter is greater than the number of `logical` parameters, `LCHOOSE` returns the last `logical` parameter listed. In the example, if the value of `Pot1` is .5, 1, or 1.4, it returns the value of `Switch1`. If `Pot1` is 2.0 or 2.8, `LCHOOSE` returns the logical value `TRUE`. If `Pot1` is 3.0 or higher, `LCHOOSE` returns the value of Pb1.
NCHOOSE	Syntax	`NCHOOSE (`*`numeric`*`, `*`numeric1`*`, `*`numeric2`*`, `*`numeric3`*`,...)`
	Example	`NCHOOSE(Pot1, Pot2, Pot3, 14.3)`
	Description	Returns the `numeric`*`x`* parameter corresponding to the integer portion of the `numeric` parameter. If the `numeric` parameter is less than 2.0, `NCHOOSE` returns `numeric1`. If the `numeric` parameter is greater than the number of `numeric`*`x`* parameters, `NCHOOSE` returns the last `numeric`*`x`* parameter listed. See the description of the *LCHOOSE* example.
TCHOOSE	Syntax	`TCHOOSE (`*`numeric`*`, `*`text1`*`, `*`text2`*`, `*`text3`*`,...)`
	Example	`TCHOOSE(Pot1, “Auto”, “Manual”, “Local”, “Locked”)`
	Description	Returns the `text` parameter corresponding to the integer portion of the `numeric` parameter. If the `numeric` parameter is less than 2.0, `TCHOOSE` returns `text1`. If the `numeric` parameter is greater than the number of `text` parameters, `TCHOOSE` returns the last logical parameter listed. See the `description` of the *LCHOOSE* example.

Mathematical Functions

ABS	Syntax	`ABS(`*`numeric`*`)`
	Example	`ABS(Pot1 – 50.0)`
	Description	Returns the absolute value of *numeric*. In the example, if Pot1 is zero, the function returns 50.0.
EXP	Syntax	`EXP(`*`numeric`*`)`
	Example	`EXP(Pot1)`
	Description	Returns the base of the natural logarithm, *e* (2.71828182), raised to the power of *numeric*. In the example, if Pot1 is 2.0, the function returns 2.71828182^2 or approximately 7.38906. EXP is the inverse of the function, LN. To calculate the values of other powers, use the exponentiation operator (^).
FACT	Syntax	`FACT(`*`numeric`*`)`
	Example	`FACT(4.2)`
	Description	Returns the factorial of the integer portion of *numeric*. In the example, the function returns the factorial of 4, or 1*2*3*4, or 24. The factorial of zero, FACT(0), or any number less than zero is one (1).
INT	Syntax	`INT(`*`numeric`*`)`
	Example	`INT(4.2)`
	Description	Rounds *numeric* down to the nearest integer. In the example, the INT function returns 4. Notice also that `INT(–8.5) = –9`. See also *TRUNC*.
LN	Syntax	`LN(`*`numeric`*`)`
	Example	`LN(PLC.AI1)`
	Description	Returns the natural logarithm of *numeric*. In the example, if PLC.AI1 = 1000.0, the function returns 6.90776.
LOG	Syntax	`LOG(`*`numeric1, numeric2`*`)`
	Example	`LOG(Pot1,2)`
	Description	Returns the logarithm of *numeric1* to the *numeric2* base, where *numeric1* is a positive real number. In the example, if Pot1 = 8.0, the function returns 3.0.

LOG10	Syntax	`LOG10(`*`numeric`*`)`
	Example	`LOG10(Pot1)`
	Description	Returns the base-10 logarithm of *numeric*, where *numeric* is a positive real number. In the example, if Pot1 = 100.0, the function returns 2.0.
MOD	Syntax	`MOD(`*`numeric1, numeric2`*`)`
	Example	`MOD(50,8)`
	Description	Returns the modulus (remainder) of *numeric1* divided by *numeric2*, where *numeric2* is not equal to zero. In the example, 50 divided by 8 equals 6 with a remainder of 2. Therefore, the function returns 2.
PI	Syntax	`PI()`
	Example	`PI()`
	Description	Returns an approximation of PI: 3.1415927. Accepts no parameters.
PRODUCT	Syntax	`PRODUCT(`*`numeric1, numeric2, numeric3,...`*`)`
	Example	`PRODUCT(2,4,6,10)`
	Description	Returns the product of all the numerics. In the example, the function returns 2*4*6*10 or 480.
RAND	Syntax	`RAND()`
	Example	`RAND() * Pot1`
	Description	Generates a new random number between zero and one every time the formula that this function is a part of is recalculated. Accepts no parameters. In the example, every time the value of Pot1 changes, the function generates a new random number and multiplies it by the value of Pot1.
ROUND	Syntax	`ROUND(`*`numeric1, numeric2`*`)`
	Example	`ROUND(Pot1,2)`
	Description	Rounds the value of *numeric1* to *numeric2* decimal places. In the example, if Pot1 equals 15.745, the function returns 15.75. If *numeric2* equals zero, the function returns an integer. If *numeric2* is less than zero, the function returns zero.

SIGN	Syntax	`SIGN(`*`numeric`*`)`
	Example	`SIGN(Pot1)`
	Description	Returns 1 if *numeric* is positive, 0 if *numeric* is 0, –1 if *numeric* is negative. In the example, if Pot1 is –0.00001, the function returns –1.
SQRT	Syntax	`SQRT(`*`numeric`*`)`
	Example	`SQRT(ABS(Pot1))`
	Description	Returns the square root of *numeric*, where *numeric* is a positive real number. In the example, if Pot1 is –25.0, the absolute function first converts the pot value to positive 25 and the square root function calculates $\sqrt{25} = 5$.
TRUNC	Syntax	`TRUNC(`*`numeric`*`)`
	Example	`TRUNC(8.9)`
	Description	Truncates *numeric* to its integer component by removing its fractional part. In the example, the TRUNC function returns 8. Notice also that `TRUNC(-8.9) = -8`. See also *INT*.

Statistical Functions

AVG	Syntax	AVG(*numeric1, numeric2, numeric3,...*)
	Example	AVG(2,4,–6,12)
	Description	Returns the arithmetic mean (average) of all the *numerics* listed. This function requires at least two numeric parameters. In the example, the function returns (2+4–6+12) / 4 or 3.0.
MAX	Syntax	MAX(*numeric1, numeric2, numeric3,...*)
	Example	MAX(2,4,–6,12)
	Description	Returns the highest value of all the *numeric* values listed. This function requires at least two numeric values. In the example, the function returns 12.
MIN	Syntax	MIN(*numeric1, numeric2, numeric3,...*)
	Example	MIN(2,4,–6,12)
	Description	Returns the lowest value of all the *numeric* values listed. This function requires at least two numeric values. In the example, the function returns –6.
STDEV	Syntax	STDEV(*numeric1, numeric2, numeric3,...*)
	Example	STDEV(2.9,4.5,5.0,4.3,3.8)
	Description	Returns the sample standard deviation of the *numeric* values listed. (Standard deviation is a measure of dispersion, calculated as the positive square root of the variance.) This function calculates the standard deviation using the non-biased or *n*–1 method. This function requires at least two numeric values. In the example, the function returns 0.796869.
STDEVP	Syntax	STDEVP(*numeric1, numeric2, numeric3,...*)
	Example	STDEVP(2.9,4.5,5.0,4.3,3.8)
	Description	Returns the standard deviation of a full population of *numeric* values. This function calculates the standard deviation using the biased or *n* method. This function requires at least two numeric values. In the example, the function returns 0.712741.

SUM	Syntax	SUM(*numeric1, numeric2, numeric3,...*)
	Example	SUM(2,4,–6,12)
	Description	Returns the sum of all the *numeric* values listed. This function requires at least two numeric values. In the example, the function returns 12.
VAR	Syntax	VAR(*numeric1, numeric2, numeric3,...*)
	Example	VAR(2.9,4.5,5.0,4.3,3.8)
	Description	Returns the sample variance of the *numeric* values listed. (Variance is a measure of dispersion.) This function calculates the variance using the non-biased or *n*–1 method. This function requires at least two numeric values. In the example, the function returns 0.635.
VARP	Syntax	VARP(*numeric1, numeric2, numeric3,...*)
	Example	VARP(2.9,4.5,5.0,4.3,3.8)
	Description	Returns the population variance of the *numeric* values listed. This function calculates the variance using the biased or *n* method. This function requires at least two numeric values. In the example, the function returns 0.508.

Text Functions

EXACT	Syntax	EXACT(*text1, text2*)
	Example	EXACT(*TextEntry1*, “batch a”)
	Description	Returns TRUE if *text1* exactly matches *text2*, otherwise returns FALSE. This function is case sensitive. In the example, the function returns TRUE if the text result of TextEntry1 exactly matches `batch a` (note that the entry in this example is all lowercase).
FIND	Syntax	FIND(*text1, text2, numeric*)
	Example	FIND(“ON”,”Reactor A is ON”,1)
	Description	Searches for *text1* within *text2* starting after *numeric* characters, and returns the position of the first character where the match begins. This function is case sensitive. The output of this function is numeric. It returns 0 if no match is found. In the example, the function searches the entire string for the word, `ON` and returns 14, indicating the position of the first character of the word. Refer also to the *SEARCH* function.
FIXED	Syntax	FIXED(*numeric1, numeric2*)
	Example	“The flow is” & FIXED(Pot1,2)
	Description	Rounds the value of *numeric1* to *numeric2* decimal places and then converts *numeric1* to a text string. In the example, if the value of Pot1 equals `123.456`, the `FIXED` function returns the text value `123.46`. Thus, the entire text string would read `The flow is 123.46`. If *numeric2* is negative or omitted, *numeric1* is rounded to the nearest whole number. For example, `FIXED(123.588)` returns `124`. See also *TEXT*.
LEFT	Syntax	LEFT(*text, numeric*)
	Example	LEFT(“Reactor A is ON”,9)
	Description	Retrieves the specified number of characters from the lefthand end of *text* and outputs it as a text value. In the example, the function returns `Reactor A`.

LEN	Syntax	LEN(*text*)
	Example	LEN(“Reactor A is ON”)
	Description	Returns the length of the text string (the number of characters in *text*). In the example, the function returns the numeric value 15.
LOWER	Syntax	LOWER(*text*)
	Example	EXACT(LOWER(*TextEntry1*),“batch a”)
	Description	Converts *text* to all lower case. In the example, the LOWER function ensures that a match is found whether TextEntry1 content reads Batch A, BATCH A, or batch A.
MID	Syntax	MID(*text*, *numeric1*, *numeric2*)
	Example	MID(“Reactor A is ON”,9,7)
	Description	Retrieves *numeric2* characters from *text*, beginning at character *numeric1*. In the example, the function returns the text value A is ON.
PROPER	Syntax	PROPER(*text*)
	Example	PROPER(“reactor A is ON”)
	Description	Capitalizes the first character of each word in *text*. In the example, the function returns the text value Reactor A Is On.
REPLACE	Syntax	REPLACE(*text1*, *numeric1*, *numeric2*, *text2*)
	Example	REPLACE(“Reactor A is ON”,9,1,“B”)
	Description	Replaces *numeric2* characters with *text2* beginning with character *numeric1* in *text1*. This function requires that the number of characters in *text2* matches the value of *numeric2*. In the example, the function returns the text value Reactor B is ON.
REPT	Syntax	REPT(*text*, *numeric*)
	Example	REPT(“Lookout”,Pot1)
	Description	Repeats *text numeric* times. In the example, if the value of Pot1 is 8, the function returns the text value, Lookout Lookout Lookout Lookout Lookout Lookout Lookout Lookout.

RIGHT	Syntax	RIGHT(*text, numeric*)
	Example	RIGHT(“Reactor A is ON”,7)
	Description	Retrieves the specified number of characters from the righthand end of *text* and outputs it as a text value. In the example, the function returns the text value A is ON.
SEARCH	Syntax	SEARCH(*text1, text2, numeric*)
	Example	SEARCH(“on”,”Reactor A is ON”,1)
	Description	Finds *text1* within *text2* beginning at *numeric* character, and returns the position of the first character where the match begins. This function is not case sensitive. The output is numeric. It returns 0 if no match is found. In the example, the function searches the entire string for the word ON, on, On, or oN and returns 14, indicating the position of the first character of the word. Refer also to *FIND*.
TEXT	Syntax	TEXT(*numeric, text*)
	Example	TEXT(Pot1,“0.00”)
	Description	Like FIXED, the TEXT function converts *numeric* to a textual value. While FIXED allows you to specify the number of decimal points, TEXT allows you specify a desired numeric format in the text parameter. If Pot1 equals 12.3456, the function would return the textual value 12.35. See *Numeric Formats* in Chapter 5, *Developer Tour*. See also *FIXED*.
TRIM	Syntax	TRIM(*text*)
	Example	TRIM(“Reactor A is ON”)
	Description	Trims multiple spaces from between words. In the example, there are multiple spaces on either side of the word, *A*. This function returns Reactor A is ON, eliminating all repeated spaces.
UPPER	Syntax	UPPER(*text*)
	Example	EXACT(UPPER(TextEntry1),“BATCH A”)
	Description	Converts *text* to all capital letters (upper case). In the example, the UPPER function ensures that a match is found whether TextEntry1 content reads Batch A, BATCH A, or batch A.

Trigonometric Functions

ACOS	Syntax	`ACOS(`*`numeric`*`)`
	Example	`ACOS(Pot1)`
	Description	Returns the arccosine of *numeric* (that is, it returns the angle of the cosine you specify as *numeric*). *Numeric* must range between –1 and 1. The result is output in radians and ranges from 0 to π. In the example, if Pot1 = 0.5, the function returns 1.0472 radians (60 degrees). You can express the arccosine in degrees by multiplying the result by 180/π.
ASIN	Syntax	`ASIN(`*`numeric`*`)`
	Example	`ASIN(Pot1)`
	Description	Returns the arcsine of *numeric* (that is, it returns the angle of the sine you specify as *numeric*). *Numeric* is the sine of the angle and must range between –1 and 1. The resulting value is given in radians and ranges from –π/2 to π/2. In the example, if Pot1 = –1, the function returns –1.5708.
ATAN	Syntax	`ATAN(`*`numeric`*`)`
	Example	`ATAN(Pot1)`
	Description	Returns the arctangent of *numeric* (that is, it returns the angle of the tangent that you specify as *numeric*). The resulting value is given in radians and ranges from –π/2 to π/2. In the example, if Pot1 = 180, the function returns 1.56524 radians (about 90 degrees).
ATAN2	Syntax	`ATAN2(`*`numericX, numericY`*`)`
	Example	`ATAN2(5, 5)`
	Description	Returns the arctangent of the specified x- and y-coordinates (that is, it returns the angle of a line extending from the origin (0,0) to a point specified by the *numericX*, *numericY* coordinate pair that you specify). The resulting value is given in radians and ranges from greater than –π to π. In the example, the function returns 0.785398 radians (45 degrees).

COS	Syntax	`COS(`*`numeric`*`)`
	Example	`COS(Pot1)`
	Description	Returns the cosine of *numeric* where *numeric* is the angle in radians. In the example, if Pot1 = 1.047, the function returns 0.500171. You convert degrees to radians by multiplying by π/180.
SIN	Syntax	`SIN(`*`numeric`*`)`
	Example	`SIN(Pot1)`
	Description	Returns the sine of *numeric* where *numeric* is the angle in radians. In the example, if Pot1 = 3.14159, the function returns 1.2246E–16 (effectively zero). You convert degrees to radians by multiplying by π/180.
TAN	Syntax	`TAN(`*`numeric`*`)`
	Example	`TAN(Pot1*PI()/180)`
	Description	Returns the tangent of *numeric* where *numeric* is the angle in radians. In the example, the value of Pot1 is 45, but it is in degrees, not radians. Convert it to radians by multiplying it by π/180. In this example, the function returns 1.

Date/Time Functions

NOW	Syntax	NOW(*logical*)
	Example	NOW($Keyboard.F1)
	Description	Returns a numeric value representing the current system date and time when any signal within the parentheses change. The result is a floating point number in which the integer represents the date and the fraction represents the time of day. In the example, when you press the function key F1, the date and time are output as a single number, like 34738.3. If you change the **Numeric Format** of the number to mm/dd/yy hh:mm:ss, the same number would be shown as 02/08/95 07:49:02.
TODAY	Syntax	TODAY(*logical*)
	Example	TODAY($Keyboard.F1)
	Description	Returns a numeric value representing the current system date when any signal within the parentheses change. The result is an integer that represents the number of days that have passed since Jan. 1, 1900. In the example, when you press the function key F1, the date is output as a single number, such as 34738. If you change the **Numeric Format** of the number to mm/dd/yy, the same number would be shown as 02/08/95.

☞ **Note** ***If you want to display the current time only, subtract the*** *today* ***function from the*** *now* ***function.***

Example: NOW($Keyboard.F1) – TODAY($Keyboard.F1)

In this example, if you want the result to update itself every second, replace $Keyboard.F1 ***with a one second pulse timer.***

8

Graphics

This chapter describes adding static and dynamic graphics to a control panel and creating and using custom graphics.

Any visible item on a Lookout control panel is a graphic. All graphics are either static or dynamic. *Static* graphics never change state, but *dynamic* graphics change state to represent process variations. Lookout provides an extensive graphics library. These graphics range from switches, potentiometers, and pushbuttons to bar graphs, valves, tanks, pumps, plates, insets, scales, and more. However, there might be times when the standard Lookout graphics do not exactly fit your needs. You can use the third-party drawing package included with your copy of Lookout, or any other drawing software, to create your own custom graphics.

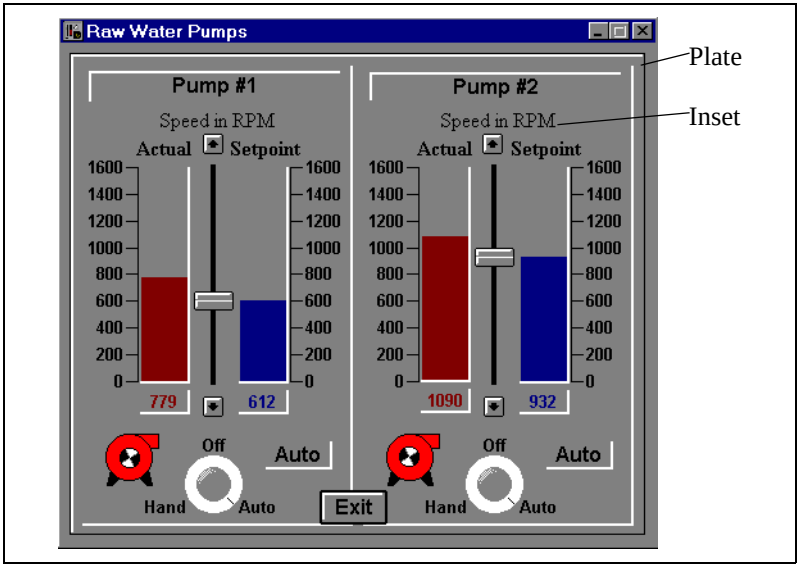
☞ **Note** ***Consider screen resolution when creating display panels (VGA vs. Super VGA). The same panel appears differently on computers using different resolution display drivers. Please read about screen resolutions in the description of*** *Panel* ***objects in Chapter 18,*** *Object Class Reference****, before designing your panels.***

Static Graphics

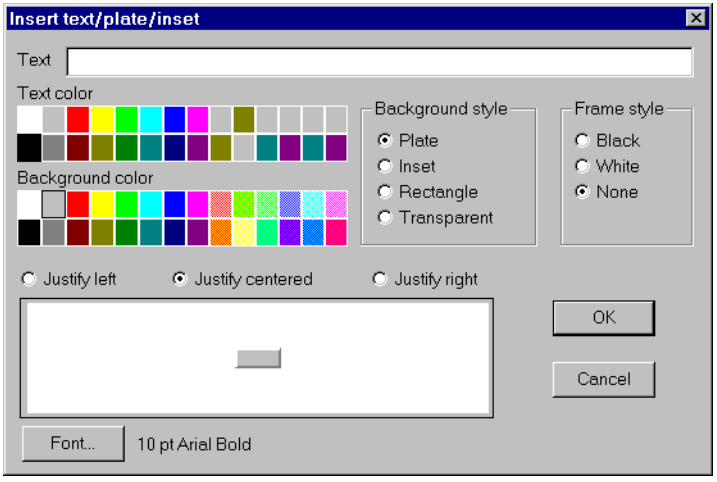
Static graphics never change state. They exist on a control panel much the same way a picture hangs on your wall, never changing or moving. Static graphics range from text labels, plates, insets, and scales to complex schematic overviews and scanned photographic images.

Displaying Text, Plates, Insets, Rectangles, and Lines

Effective use of text, plates, and insets make control panels intuitive and easy to use. The example below demonstrates how plates and insets organize information about two pumps.



To create static text, plates, insets, rectangles or lines in Lookout, select the **Insert»Text/plate/inset…** command. The following dialog box appears.



If you want to create a rectangle, line, plate or inset, leave the **Text** field blank and choose a **Background style**. The preview window displays the element as it appears on a control panel.

To create a vertical or horizontal line, choose **Rectangle**. After clicking on **OK**, size the rectangle to the desired width and length. If you need the rectangle an exact size, remember to use the yellow status bar. It indicates dimensions.

☞ **Note** ***Lookout uses color grids in many dialog boxes. The grids make selecting colors quick and visually helpful. However, they might look slightly different from computer to computer.***

Some computer display adapters can display only 16 solid colors on the screen. The standard VGA adapter in conjunction with the Windows VGA driver supports only 16 colors directly. Windows uses a technique called *dithering* to simulate the display of colors not directly supported as solid colors. For example, Lookout may display orange as an alternating pixel pattern of red and yellow bits on the screen.

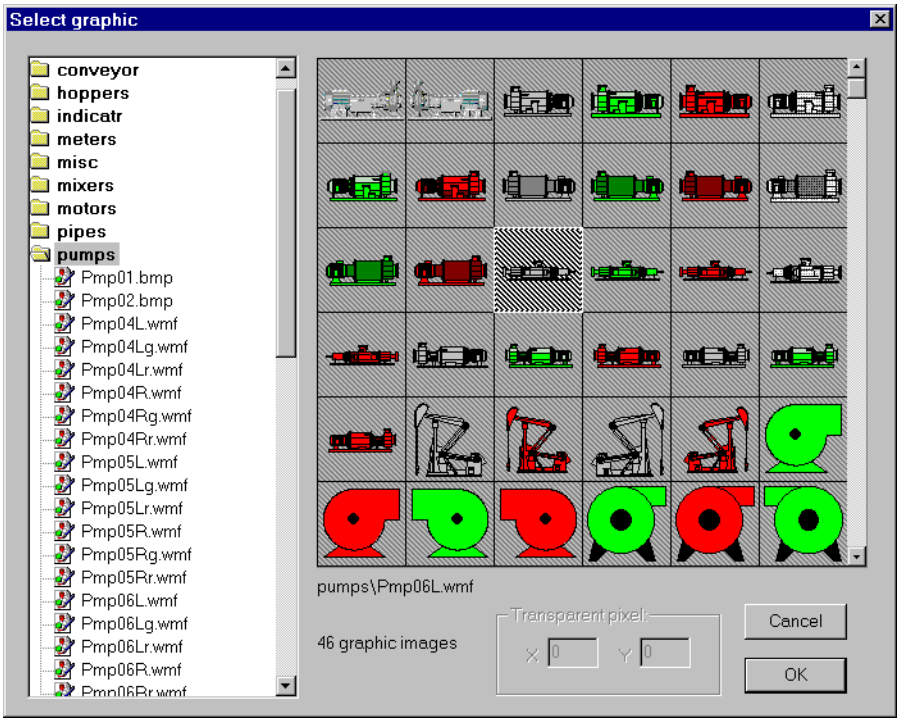
Some objects in Lookout require solid colors, so if you specify a dithered color, Lookout uses the nearest solid color instead. For example, a Trend object requires a solid color for its background and solid colors for the trend lines. Furthermore, Lookout always displays text in a solid color. If you specify pastel green for a trend line color, you might get yellow instead. Bar graphs and control panel backgrounds can display dithered colors.

☞ **Note** ***Some display adapters support 256 or even 16 million colors on the screen at one time. Even though Lookout allows you to specify only 28 colors, Lookout displays custom graphic files on control panels using all available colors.***

Displaying Static Custom Graphics

Lookout supports two graphical file types—Windows Device-Independent Bitmap (`.BMP`) and Windows Metafile (`.WMF`).

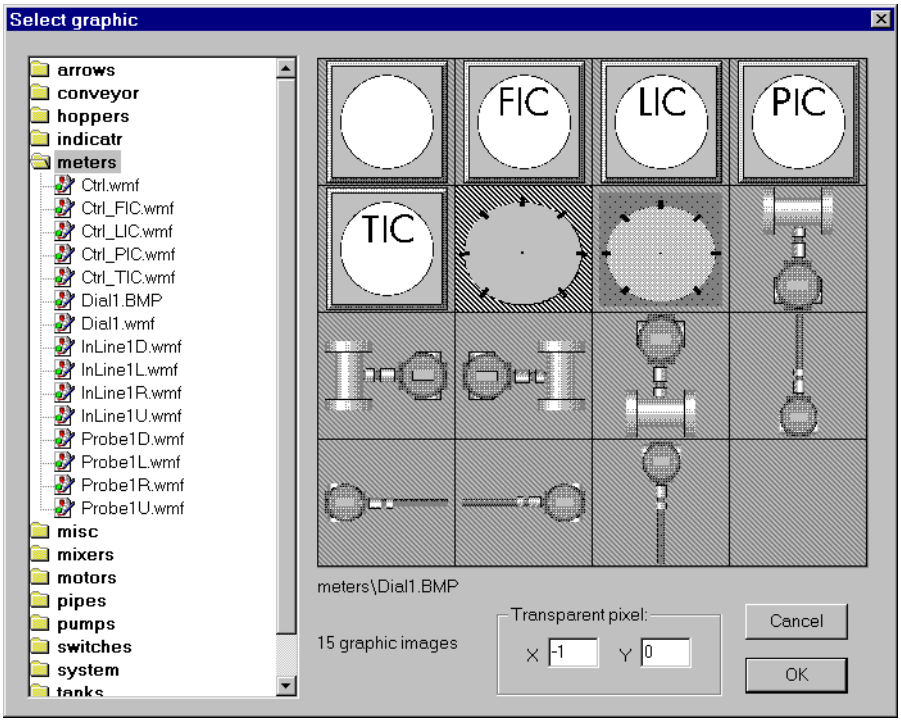
To display static bitmap and metafile graphic files on a control panel, select the **Insert»Graphic…** command. The **Insert graphic** dialog box appears.



From the Insert graphic dialog box, you can scroll through various directories containing a variety of graphic files. The list box includes all bitmaps and metafiles located in the `GRAPHICS` subdirectory under the root `LOOKOUT` directory. As you select each category, you see thumbnail sketches displayed in the preview window to the right of the list box. Because graphics might stretch or shrink to fit in the preview window, they might not appear exactly as they do on a control panel.

☞ **Note** ***If you know the name of the file, type the first letter of the filename. The list box automatically scrolls to the first file beginning with that letter.***

If you choose a bitmap (`.BMP`) file, you can use the **Transparent pixel** data fields to specify which color pixels in that graphic you want to be transparent in Lookout. Imagine the viewing area as an X-Y coordinate plane with 0,0 being the top left corner of the graphic (not the entire viewing area). You can enter any **X** and **Y** coordinates, and the corresponding pixels become transparent, as do all other pixels that color. The **Transparent pixel** fields do not have any default values. If you leave the **X** and **Y** fields blank, no pixels become transparent.



You can use a multicolor bitmap as a type of mask. For this to work, inside of the bitmap (the gray area) must be transparent and the rest of the graphic opaque—masking the underlying part of the control panel. You can insert another Lookout element, such as a bar graph, to furnish a convenient visual warning, such as a rising color level.

You could guess the X,Y coordinates of any grey pixel on the graphic. It helps to know that the center of the graphic is `-1,-1`. Because `-1,-1` is a black pixel in this example, you would have to offset your choice to `-1,0`, (for example). Because this is a gray pixel, all other gray pixels become transparent when inserted on a control panel.

Note ***Windows metafiles are normally easier to use and manipulate than bitmap images. Unlike bitmaps, metafiles can be resized in Lookout. Because of their inherent structure, you can also use metafiles as masks without specifying transparent pixels. Had the example above used a .WMF graphic, the area in gray would have appeared in the Lookout window as a crosshatched area.***

Dynamic Graphics

Many process control panels require some type of animation, such as a pump changing colors to represent on and off. Table 8-1 lists the Lookout tools that make use of animation.

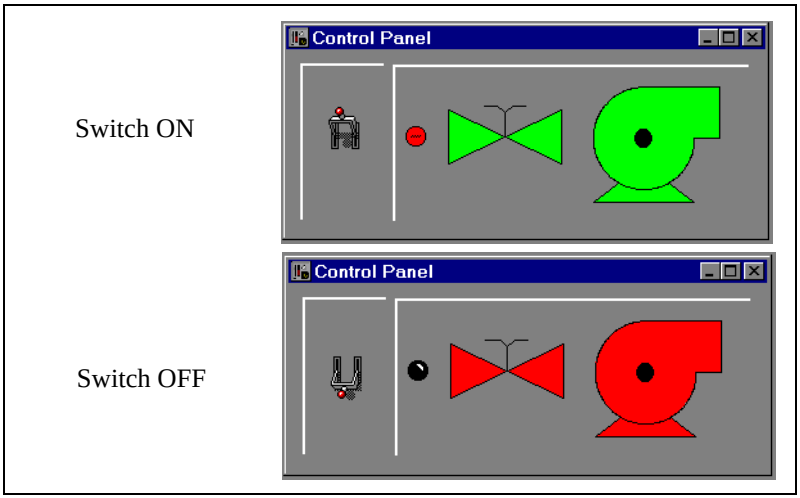
Table 8-1. Tools for Displaying Dynamic Graphics

Object Class	Description
Animator object class	An Animator object provides full graphical animation including horizontal and vertical motion, dynamic resizing and visibility, dynamic sequencing, and color changing.
Multistate object class	A Multistate object displays up to six different custom graphics based on advanced if-then-else logic.
Pipe object class	A Pipe object makes a rectangle or line (of any dimension) change colors or blink, based on advanced if-then-else logic.
DialGauge object class	A DialGauge object displays a numeric signal as a sweeping needle on an analog gauge or dial.
Gauge object class	A Gauge object makes a numeric expression (digital number or barchart) change colors or blink based on advanced logic.
Spinner object class	A Spinner object is a small rotating disk. It can be turned on and off with a logical signal, and its rotation speed and direction are controlled by a numeric value.
Switch object class	A Switch object can represent its two positions using standard switch symbols or custom graphics.
Pushbutton object class	A Pushbutton object looks like a button that changes when depressed, but you can also make it transparent. You might use transparent pushbuttons over custom graphics.
Pot object class	A Pot object can be displayed as a knob, vertical slider, horizontal slider, increment and decrement buttons, or a digital number.
Logical expression	When you create an expression that results in a logical value, you can represent that value using standard lights, text, or custom graphics.
Numeric expression	When you create an expression that results in a numeric value, you can represent that value using a digital number or a vertical or horizontal bar chart.
Text expression	When you create an expression that results in a text value, you can represent that value using any style and size font loaded on your computer.

For information on a specific object class, refer to Chapter 18, *Object Class Reference*. For information on expressions, refer to Chapter 7, *Expressions*.

Displaying Dynamic Logical Signals

Logical expressions and Switch objects are two commonly used graphical animation tools. You can display the signal a Switch object generates in a variety of ways.

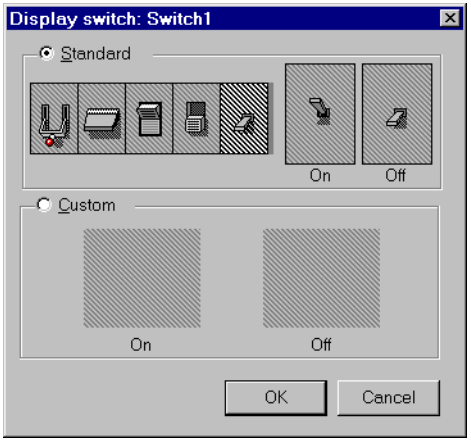


The Switch is shown in both the On and Off positions. The graphics on the right side of the control panel show three ways of graphically displaying the logical signal generated by the Switch.

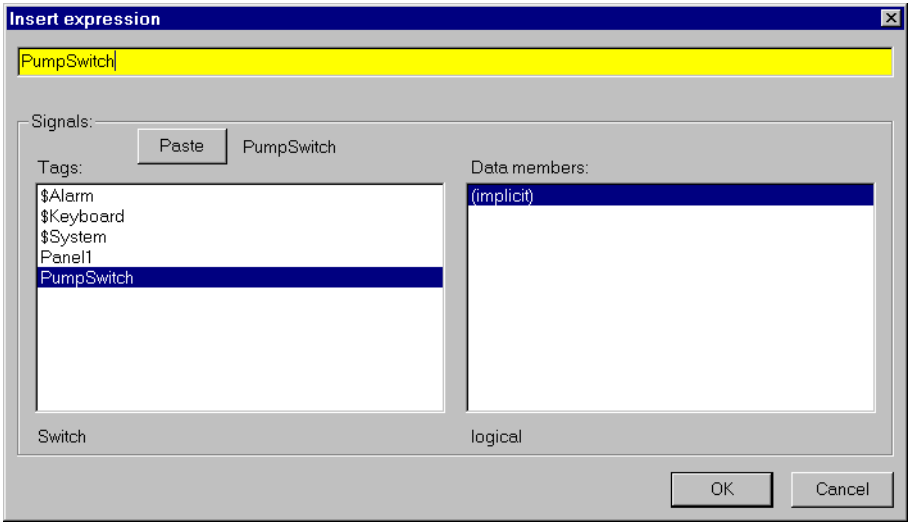
Notice the difference between the graphics representing the Switch *object* and the graphics representing the Switch *signal*. The graphics representing the Switch signal are *expressions* of the `(implicit)` value of the switch, created through the **Insert»Expression…** menu command.

To create a Switch object, first choose the logical signal that you want to represent with dynamic graphics (for example, a logical input from a PLC). Then select an object to display the logical signal. For this example, use a Switch to simulate a contact from a motor starter relay.

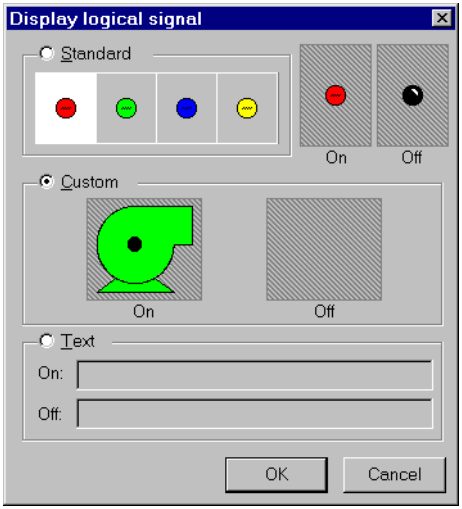
Create a Switch and call it `PumpSwitch`. After you define the Switch parameters, Lookout presents the display parameters dialog box for the Switch object. You can choose to represent `PumpSwitch` with one of the standard graphics, or you can use custom graphics. For this example, select a standard graphic and click on **OK**.



To insert an expression representing the signal generated by the Switch object, use the **Insert»Expression…** command and select `PumpSwitch`.



When you click on **OK**, Lookout presents the **Display logical signal** dialog box. From the **Display logical signal** dialog box, select display characteristics for a logical signal expression.



Click on the **Custom** selection, and then on the **On** and **Off** list boxes, scrolling through the choices until you find the appropriate graphic. Your **On** and **Off** graphic selections appear as thumbnail sketches in the two preview windows to confirm and verify your selection.

Click on **OK** and test your example by flipping the switch. The graphics should change according to your selections in the **On** and **Off** preview windows. To customize your switch, experiment with the **Text** selection in the **Display logical signal** dialog box.

Use this same basic method to display any dynamic signal on a control panel. If you want to represent more than two conditions with dynamic graphics, use a Multistate object. Similarly, you can use a Pipe object to make a line or rectangle dynamically change colors and blink.

Displaying Dynamic Numeric Signals

This section shows how to use a Pot object to display dynamic numeric signals. Figure 8-1 shows the Pot as a slider, a digital entry, and increment and decrement buttons—all are graphical representations of the same Pot object. The graphics to the right and below the control panel are two ways of graphically displaying the numeric signal generated by the Pot.

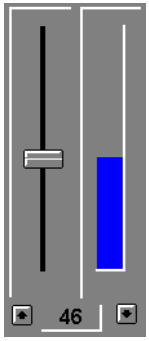
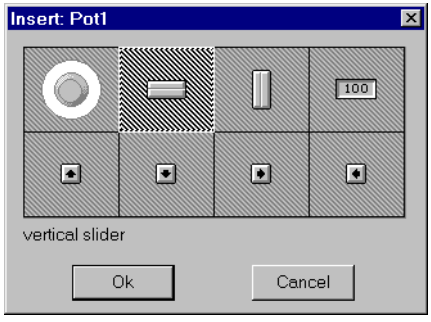
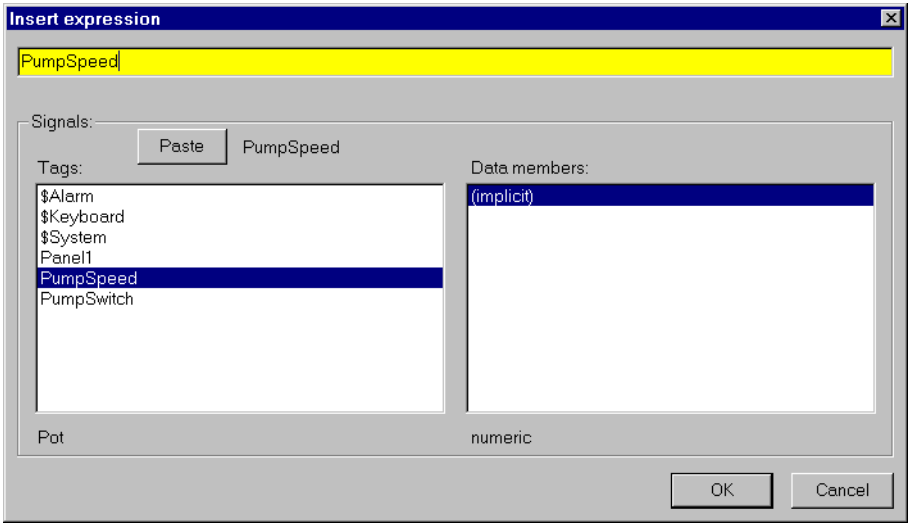


Figure 8-1. Pot Object as a Slider, a Digital Entry, and Increment and Decrement Buttons

To use a Pot object to display a signal, first choose the numeric signal that you want to represent with dynamic graphics, such as an analog input from a PLC. Then select an object to graphically display the dynamic numeric signal. For this example, create a Pot object and name it `PumpSpeed`. After you define the Pot parameters, Lookout presents its display parameters dialog box. In this example, the dialog box is named **Insert: PumpSpeed**. You can choose to represent the Pot with any of the standard graphics.

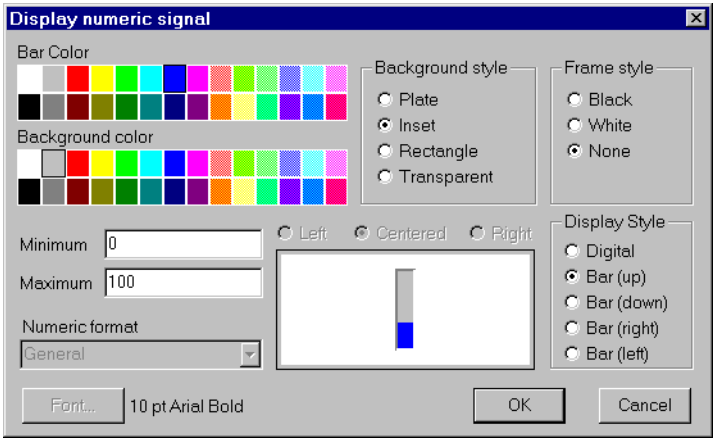


Use the **Insert»Expression…** command and select `PumpSpeed` to insert an expression that represents the signal generated by the Pot object.



When you click on **OK**, Lookout presents the **Display numeric signal** dialog box. From this dialog box, select the display characteristics for a numeric signal expression.

If you choose **Digital** display style, you can use the **Font** button to select the desired font style and size. You can also specify a **Numeric format** for the value. For more information on numeric formats, see Chapter 5, *Developer Tour*.



Click on **OK** and test your example by adjusting the pot. The graphics should change according to your selections.

If you want to represent the numeric signal with moving custom graphics, use the Animator object class.

Displaying Dynamic Text Signals

You can display dynamic text messages on a control panel with text expressions. You can easily display up to two separate text messages using logical signals, but there are times when you need to display three or more separate text messages in a single statement. If you do have more than two separate messages, a numeric signal might determine which message to display.

To graphically display a dynamic text signal, first choose the numeric signal that you want to use to control which message is displayed, perhaps an analog input from a PLC. Then select a graphical object to display that signal. This example uses a Pot object.

Create a Pot object and call it `Pot1`. Define the Pot minimum, maximum, and resolution parameters as 1, 4, and 1 respectively. Select the **Insert»Expression…** command and enter the expression from Figure 8-2. See Chapter 7, *Expressions*, for information on the *TCHOOSE* function.

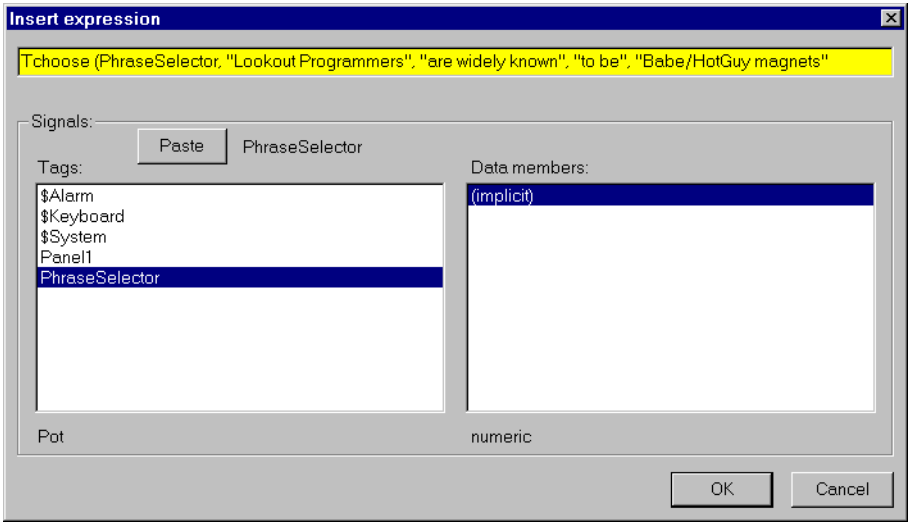
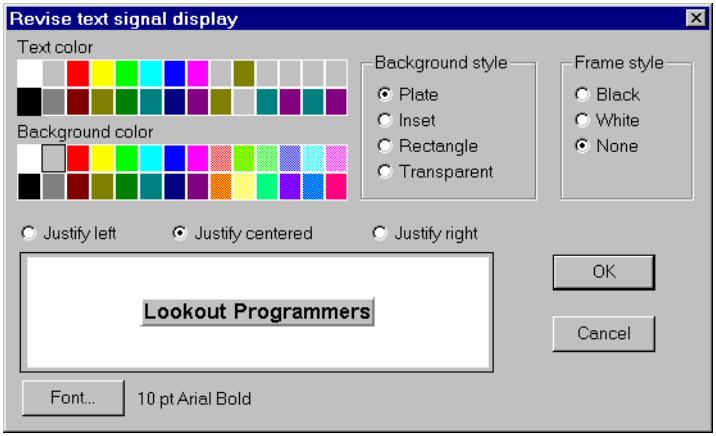


Figure 8-2. Using Expressions To Display Dynamic Text Signals

When you click on **OK**, Lookout presents the **Display text signal** dialog box. From this dialog box, select display characteristics for a text signal expression. Use **Inset** as your background style.



Click on **OK** and test your example by adjusting the pot. The text should change according to your expression.

Creating Custom Graphics

There may be times when none of the Lookout standard graphics exactly fit your needs. To meet those needs, National Instruments includes a third-party drawing package with every copy of Lookout. You can use it, or any other drawing software, to create your own custom graphics. Lookout supports both Windows Device-Independent Bitmaps (`.BMP`) and Windows Metafiles (`.WMF`).

After you create a custom graphic file, copy it to the `GRAPHICS` subdirectories in your `LOOKOUT` root directory. Because Lookout can use only graphics located in the `GRAPHICS` directory and subdirectories, keep all standard and custom graphics in the `GRAPHICS` subdirectories. Lookout can access your custom graphic file any number of times in the same or different process files from the `GRAPHICS` subdirectories.

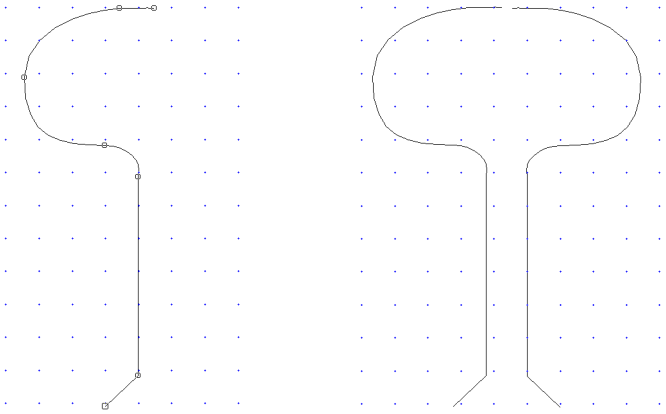
Step-by-Step Example: Creating Custom Graphics

This section contains an example demonstrating the creation of custom graphics for use in Lookout. This example sketches out the creation of an elevated tank graphic you can use to show water level in a real tank. You export that tank graphic to Lookout, implement the display in Lookout, and test it. The example may only parallel what you would have to do with your own graphics creation program, but should serve as an illustration of the important points.

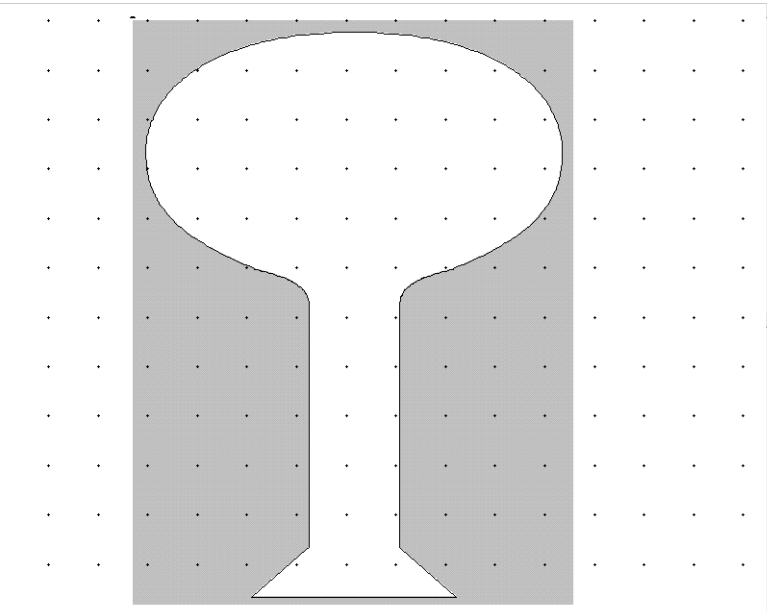
If you have not already mastered using a drawing program, you should allow yourself time to become accustomed to using the third party software included with your Lookout kit, or to using the drawing application of your choice.

Creating the Graphic

1. Draw half of the tank.
2. Copy the tank half and mirror the image to create the other half.
3. Connect and refine the halves to create an enclosed tank object.



4. Refine your image until it looks the way you want
5. Draw a rectangle around the tank and connect both the rectangle and tank into a single object.
6. To match the background color of the mask to the control panel, select the object group and click on the gray color bar selection.
7. Select invisible for the line style.

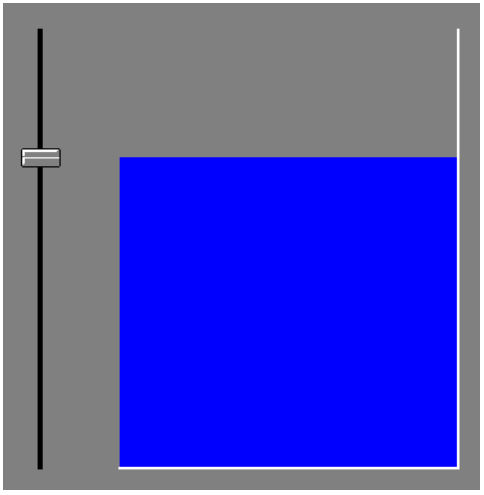


Exporting the Graphic to Lookout

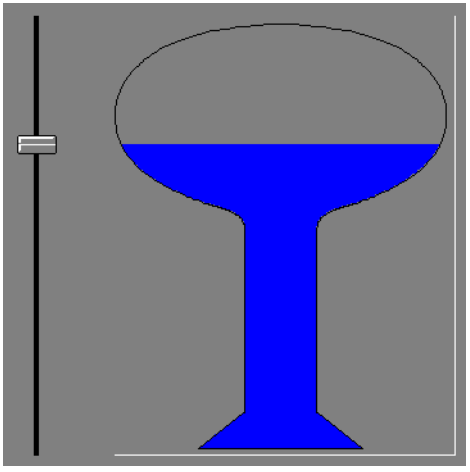
1. Export your new graphic as a `Windows Metafile` (`.WMF`).
2. Change the export directory to `C:\LOOKOUT\GRAPHICS` and name your new graphic `ELEVTANK.WMF`.
3. Export.

Testing the Graphic in Lookout

1. Launch Lookout.
2. Create a Pot object using the **Object»Create…** command.
3. Select **Insert»Expression…**, choose the Pot object, and click on **OK**.
4. In the **Display numeric signal** dialog box, select **Bar (up)** for the **Display style**, **Rectangle** for the **Background Style**, and click on **OK**.



5. Use the **Insert»Graphic…** command to insert your new graphic, `ELEVTANK.WMF`, over the bar graph.
6. Position the graphic over the bar graph and stretch it to size.
7. Toggle out of edit mode and test your example by moving the slider up and down.



Graphic File Types

Lookout accepts both Windows Device-Independent Bitmaps (.BMP) and Windows Metafiles (.WMF).

Bitmaps

Bitmaps are raster files, made up of differently colored pixels. Because each pixel in the bitmap takes up one pixel on the screen, bitmap images are always rectangular and never resizable. You can display raster images on control panels that are not rectangular in Lookout by using transparent pixels for the parts of the rectangle you do not want to show.

Bitmap files typically have a .BMP file extension. The Paint program that comes with Windows can read .PCX bitmap files (another very common bitmap format) and convert those files to Windows bitmaps.

Metafiles

Metafiles are vector files, consisting of coordinates that are connected by lines and curves, as well as area-fill commands. A vector file may consist solely of two sets of coordinates connected by a line or a complex set of colored area fills and colored lines and curves to create line art images. Metafiles images are not necessarily rectangular.

Because metafiles contain a set of coordinates, they can be resized and stretched to any size or aspect ratio. Microsoft does not add information in the basic metafile file format for metafile size information. The Aldus Corporation (authors of PageMaker) created a metafile header that contains metafile size information. Most software packages that generate metafiles also add this header information. Without the header, Lookout cannot maintain the correct aspect ratio (width to height ratio) for a metafile. To determine if a metafile has this information, resize the graphic on a control panel while holding down <Ctrl>. If aspect information is available, Lookout does not stretch the metafile drawing out of proportion.

Bitmaps or Metafiles?

Which format is better: bitmap or metafile? Both have strengths and weaknesses. You may want to use a combination of metafiles and bitmaps. Bitmaps effectively handle large background schematics or system overviews, and metafiles work well for individual pumps, valves, lamps, and other miscellaneous graphics.

Bitmaps usually display faster than metafiles. In fact, a complex drawing rendered as a bitmap may display 100 times faster than the corresponding metafile. If you want to display scanned photographic quality images with hundreds of colors, use bitmaps.

Metafiles are resizable—you can stretch them to any size or aspect ratio. Metafiles are typically smaller than bitmap files, so they take up less disk space and consume less memory than bitmap files. You can use one file that contains a metafile drawing of a pump to display several pumps of various sizes on the screen. With most drawing programs, you can save your line art images as metafiles. You can also copy the image on screen to the Windows clipboard and paste it into a paint program for bitmap conversion.

Memory Considerations

Lookout loads each graphic into computer memory the first time it is displayed. The image remains in memory until Lookout or another application needs more memory than is available. When more memory is needed, the graphic is discarded from memory and reloaded from disk the next time it is displayed, so that you can display more bitmaps and metafiles on the screen than can be held in memory at one time.

If you are running Windows in enhanced mode on a 386 or 486 computer, you have virtual memory. Windows uses virtual memory to swap memory images between RAM and disk, giving applications the appearance of more memory. If available memory becomes low and Windows must use virtual memory to handle applications and data between disk and memory, you might notice a slower application speed. For more information on virtual memory, refer to your Windows user guide. If your computer disk drive light flashes every time you pull up a new control panel in Lookout, consider purchasing more RAM.

9

Alarms

This chapter describes generated alarms and configuration services provided by the Lookout alarm subsystem. Objects and database parameters generate alarms in Lookout. As a Lookout *environment service*, the alarm subsystem filters, displays, logs, and prints alarms.

Defining Alarm Conditions

You can create an alarm condition two ways. First, you can define alarm parameters when modifying an object database (*Database-Generated Alarms*). Second, you use Alarm objects to define alarm conditions (*Alarm Object* method).

Database-Generated Alarms

Most objects contain numeric or logical data members. If they exceed specified conditions, you want Lookout to generate an alarm. These conditions can include High, Low, High-High, and Low-Low. Select **Object»Edit Database…** to modify alarm parameters for an object data member in its database dialog box (see Figure 9-1).

This method offers the most efficient way to define standard or simple alarm conditions. Furthermore, this method is especially useful when configuring alarms for data that originate from objects with large databases, like driver objects and DataTable objects. See *Editing Object Databases* in Chapter 4, *Using Lookout*, for more information.

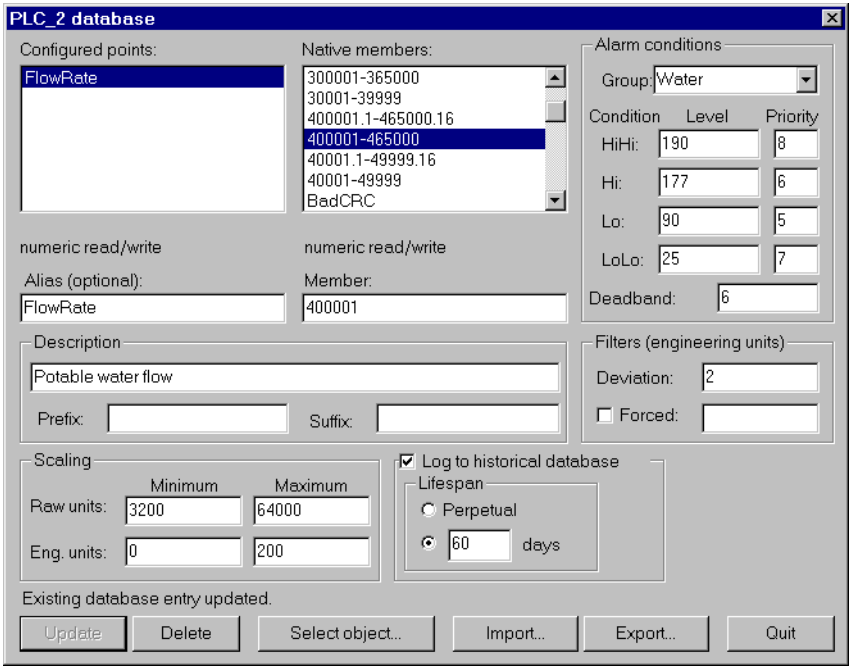


Figure 9-1. Creating Database-Generated Alarms

Alarm Objects

For dedicated Alarm objects, you define and create Alarm objects on an individual basis with the **Object»Create…** command. You can create alarms that are triggered by any event or combination of events you specify in the **Create Alarm** dialog box using as simple or complex an alarm criterion as needed. You do not have to use the traditional set of alarm conditions (that is, High, Low, Rate-of-Change). See the *Alarm* section in Chapter 18, *Object Class Reference*, for additional information.

Alarm objects are not the only source of alarm signals. Most object classes generate their own alarm signals—driver, Spreadsheet, and Expression objects can also generate alarms.

Several object classes generate *circular reference* alarms. A circular reference alarm defines a condition where the signal generated by an object is sent back to that object as a parameter, either directly or indirectly. Circular references are always priority 10 alarms, and you should correct them during process file testing.

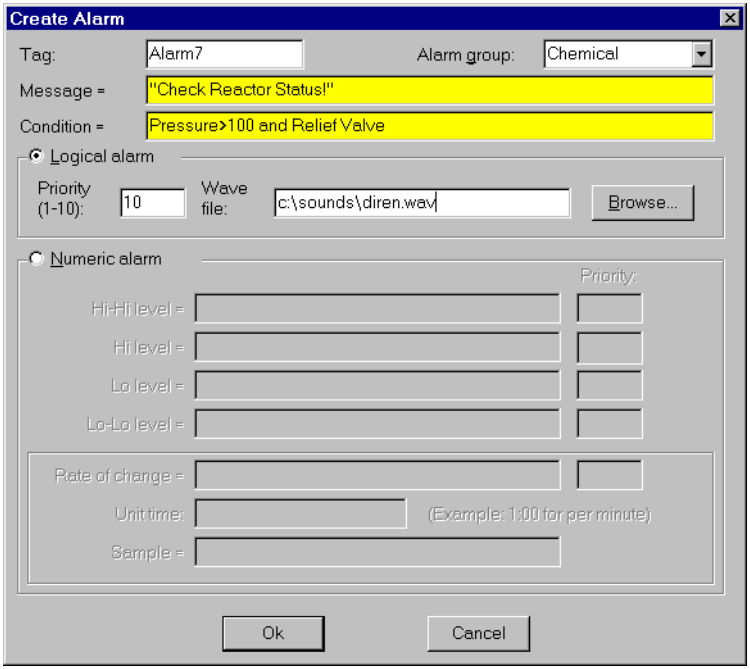


Figure 9-2. Creating an Alarm Object

The Alarm Subsystem

You use the alarm subsystem, which is comprised of several components, to control how alarms are displayed, acknowledged, printed, grouped, prioritized, and filtered. Before you implement a structure for your alarming system, you should understand these various components.

Alarm Groups

For organizational purposes, you can classify alarms into groups. Alarm groups allow operators to temporarily filter unwanted alarms and acknowledge alarms on a group-by-group basis.

You can create any number of alarm groups and assign any number of alarms to any group. However, you should carefully plan your alarming structure so operators can filter alarms by meaningful groups. With a carefully planned structure, you can handle specific groups and/or priorities if your system experiences a large number of alarms. See the *Alarm* object in Chapter 18, *Object Class Reference*, and *Editing Object Databases* in Chapter 4, *Using Lookout*, for additional information.

Alarm Priorities

Assign each alarm a priority level ranging from 1 to 10, where Priority 1 is the lowest priority alarm and Priority 10 is the highest priority alarm.

Priority 1 alarms do not require operator acknowledgment. When they are deactivated, Priority 1 alarms are removed from the Alarms window by Lookout, regardless of acknowledgment. Lookout does not permanently log Priority 1 alarms to disk.

You should assign Priority 1 to only minor, inconsequential alarms. Often, priority 1 alarms do not exist in control strategies. If an occurrence is significant enough to be classified as an alarm condition, you should assign a priority level that requires operator acknowledgment and permanent disk file logging.

Priority 2 and higher alarms require operator acknowledgment before they are removed from the alarm list. However, Lookout does not log Priority 2 alarms to disk. Assign a Priority 2 level or lower to alarms that do not need to be permanently recorded.

Priority 3 and higher alarms require operator acknowledgment before Lookout removes them from the alarm list. Lookout immediately stores Priority 3 and higher alarms to disk in binary format when they are activated, deactivated, and acknowledged.

The **Data files location** parameter in the **System Options** dialog box specifies the root directory under which all data logging should begin. Lookout stores alarm files in monthly subdirectories under this root directory. See the *Options»System* command in Chapter 16, *Runtime Menu Commands*, for more information. Priority level 3 through 10 alarms function identically and are logged to the `ALARMS.DAT` file in the data directory under the specified root directory for the current month.

☞ **Note** ***The Lookout Express package does not log alarms to disk. Rather, you can view alarms that are active—or inactive but not yet acknowledged—in the Alarms window.***

Alarms Window

The Alarms window lists all active and unacknowledged alarms. Alarms are listed in chronological order with the most recent alarm at the top of the list. If there are too many alarms to see at once, you can use the scroll bar at the right of the Alarms window to scroll through the list.

To view the Alarms window, press <Ctrl-A>, select **Alarms»Show** or click on the alarm indicator box on the far right side of the Lookout Status bar at the bottom of the screen (see Figure 9-3).

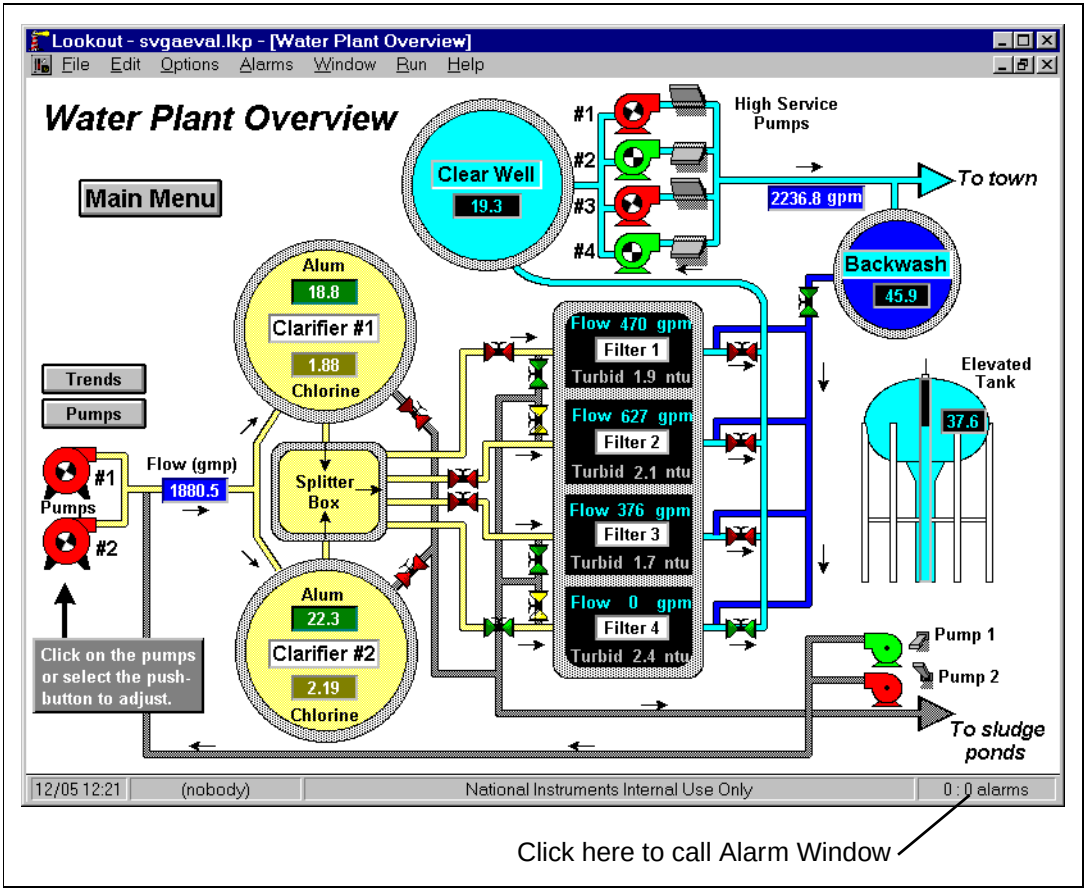


Figure 9-3. Activating the Alarms Window from the Status Bar

To quickly identify alarm statuses, use the Lookout Alarms window color scheme, as defined in Table 9-1.

Table 9-1. Alarm Colors

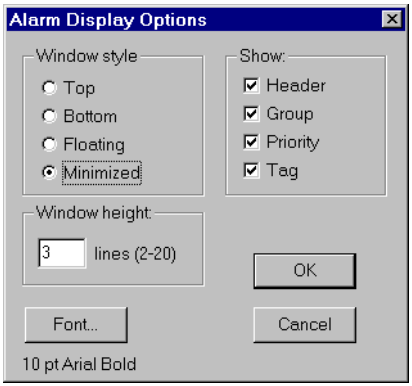
Color	Alarm Status
Red	Active
Blue	Unacknowledged, Inactive
Red and Black	Acknowledged, Active (Alarm information, including time, group, priority, and tag, appears in black and the description in red).

When you acknowledge an inactive alarm or an acknowledged alarm deactivates, Lookout removes the alarm from the Alarms window. Because a new line is added to the list every time the alarm activates, Lookout might list the same alarm condition multiple times in the Alarms window. After listing 100 instances of the same alarm, Lookout swaps the oldest alarm condition for each new occurrence. Therefore, it displays the most recent 100 occurrences of each individual alarm in the Alarms window. Other alarms continue to accumulate as expected until they reach their individual limits of 100 occurrences.

To view detailed information about a particular alarm, select the alarm in the Alarms window and use the **Alarms»Acknowledge…** command.

Alarm Display Options

With the **Alarms»Display Options…** command, you can change the display style of the Alarms window (Top, Bottom, Floating, or Minimized) and modify internal Alarms window display formats such as font, header, and other alarm information.



The **Window style** determines the position of the Alarms window in the Lookout workspace. If you select **Floating**, the Alarms window appears as a popup style control panel that you can resize and move on the screen. You can minimize a floating Alarms window at any time.

If you use either the **Top** or **Bottom** window type, the **Window height** specifies the number of alarms Lookout can display in the Alarms window. The actual height of the Alarms window adjusts automatically depending on the selected font and **Window height** setting. You can resize a floating Alarms window at any time with the sizing border. If more alarms occur than can be displayed in the Alarms window at once, a scroll bar appears along the right side of the window.

Figure 9-4 is an example of an Alarms window with **Window style** set to **Bottom**, all **Show** options selected, and **Window height** set to 3.

Figure 9-4. Example Alarms Window

Alarm Filters

Use the **Alarms»Filter Options…** command to filter the alarms by group, priority, and occurrence through the **Alarm Filters** dialog box.

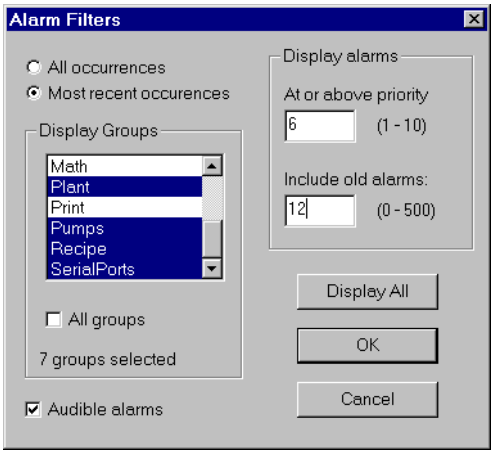


Figure 9-5. Alarm Filters Dialog Box

Most Recent Occurrences removes multiple occurrences of an alarm and displays only the latest occurrence. This feature is useful when an alarmed event continuously reoccurs and generates multiple alarm instances. **All Occurrences** displays every instance of a particular alarm (up to 100 occurrences of the same alarm).

With the **Display Alarms at or above Priority** setting, you display alarms with the specified alarm priority setting or higher and exclude all lower priority alarms.

With the **Display Groups** selections, you specify the alarm groups you want to display in the Alarms window and Lookout excludes all others. Select the groups you want to see or check the **All Groups** box to display every alarm group. You must deselect the **All Groups** box before you can choose individual **Display Groups**.

You can toggle the **Audible Alarms** option on and off. When enabled, new alarm occurrences beep on your computer speaker. After the first six beeps, the beeping frequency reduces to once every ten seconds and continues at that frequency until the alarm is acknowledged. Lookout continues to beep until you acknowledge every alarm. To quickly silence the beeping without acknowledging the alarm, press <Ctrl-S>.

The configuration shown in Figure 9-5, instructs Lookout to display the most recent alarm occurrence in the *Conveyors*, *Pipeline*, and *Plant* groups with a priority of 6 or higher. There might be active alarms occurring in other groups or in the selected groups with lower level priorities. There might be multiple occurrences of the same alarm, but these alarms are not displayed in the Alarms window because of the filter settings.

If you filter some alarms, an operator might not realize that an alarm is occurring because it is not displayed in the Alarms window. To determine how many alarms are filtered out, use the information from the right corner of the Status Bar, which displays the number of visible alarms out of the total number of alarms (visible:total). The difference between the two values is the number of filtered alarms. In Figure 9-6, there are 15 visible alarms, 25 total alarms, and 10 filtered alarms.

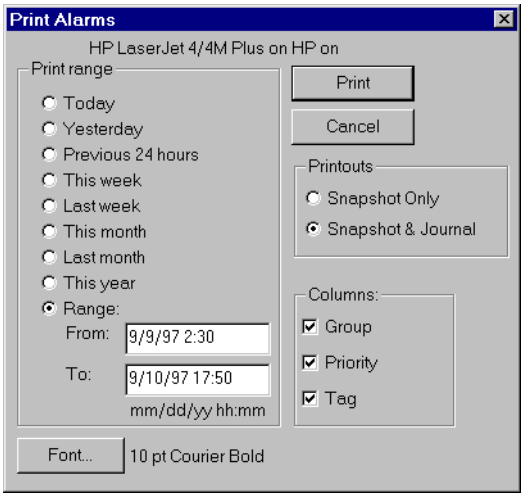
Figure 9-6. Determining the Number of Filtered Alarms from the Status Bar

To see all alarms, click on the **Display All** button in the **Alarm Filters** dialog box and then click on **OK**. The Alarms window displays all active and unacknowledged alarms. The status bar reflects the change.

Alarm Print

Note ***Because it does not log alarms to disk, the Lookout Express package does not allow printing of logged alarms.***

The **Alarms»Print…** command invokes a dialog box that you use to specify the time period of the alarms to print.



In the example above, the **Range** option specifies an arbitrary time span. By entering the exact dates and times (in military style), you can print all of the alarms logged during the specified time period. When you click on **Print**, Lookout opens the appropriate alarm file(s), formats the report according to the **Columns** settings, and sends the output to the Windows print manager (which sends the report to your printer).

The **Printouts** selections determine the exact alarm information included in the report. **Snapshot Only** provides a picture of the alarm statuses at the beginning of the specified print range but does not indicate what happened during the time span. **Snapshot & Journal** creates a printout of everything that occurred during the time span from the beginning of the range.

Alarm Acknowledgment

Before you can acknowledge an alarm, you must select it. To select an alarm, click on the alarm line in the Alarms window. A check mark appears in the far left column indicating that the alarm is selected. To deselect an alarm, click on the alarm line again and the check mark disappears. You can select and deselect any combination of alarms in this manner.

Use the **Alarms»Select All** command to select all alarms for acknowledgment. Because selecting each alarm individually can be very time consuming, this command is especially useful if your process is experiencing a high number of alarms.

Use the **Alarms»Deselect All** command to deselect all alarms that were previously selected for acknowledgment. If you want to deselect only specific alarms, click on the individual alarms you want deselected.

The **Alarms»Acknowledge…** command acknowledges alarms that you have selected for acknowledgment. If you have not selected alarms when you invoke this command, the following message box appears:

If you select one or more alarms for acknowledgment, a dialog box similar to Figure 9-7 appears to provide additional information about the alarm(s). Through this dialog box, you can acknowledge alarms on an individual basis (by clicking on **Acknowledge**) or on a global basis (by clicking on **Acknowledge All**). When you acknowledge an inactive alarm or when an acknowledged, active alarm deactivates, Lookout removes it from the Alarms window.

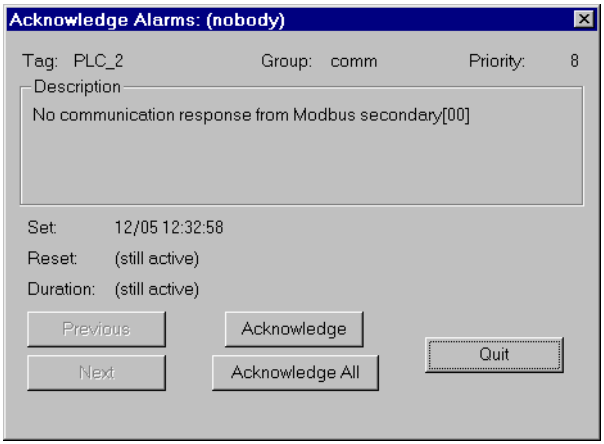


Figure 9-7. Acknowledging Alarms

The **Acknowledge All** button acknowledges all *selected* alarms—not all alarms. With the **Previous** and **Next** buttons, you can tab through all the alarms selected for acknowledgment. When you acknowledge an alarm, Lookout logs to disk the date, time, and account name of the operator that acknowledged it. Click on the **Quit** button to close the dialog box.

A special global object, $Alarm, acknowledges alarms with pushbuttons or other logical expressions. This allows you to bypass the menu commands or acknowledge alarms from remote Lookout network nodes. For instance, you could create a single pushbutton that acknowledges all alarms, or you could create multiple pushbuttons for acknowledging specific groups of alarms.

With the $Alarm object, you can access numeric signals that indicate the number of currently active alarms or the number of unacknowledged alarms in a particular group. See *$Alarm* in Chapter 18, *Object Class Reference*, for more information.

10

Security

This chapter describes Lookout accounts and the environment service security subsystem, which oversees process file security, control security, viewing security, and action verification. With this system, you selectively determine which operators control particular objects, which operators view particular control panels, and which objects prompt the operator for command verification.

Lookout requires operators to log on with a predefined name and corresponding password (if any). To log on, use the **File»Log on…** command, press <Ctrl-L>, or click on the Account name box in the status bar. Collectively, the name and password are known as an *account*. Each account has a security level. Because Lookout uses account names when logging events to disk and when operators acknowledge alarms, you can identify the operator logged on when an event occurs.

You can access the name and security level of the currently logged Lookout user through the $System object, using the `username` and `seclevel` data members.

☞ **Note** ***The Lookout Express system does not log events to disk.***

The Lookout virtual keyboard that you can use with a touch sensitive screen or a mouse is subject to the same security options.

To enable the Virtual Keyboard, select **Options»System…** and then check **Left Mouse Click** or **Right Mouse Click** in the **Virtual Keyboard Pops Up** section of the dialog box.

When this feature is enabled, clicking in a data entry field or touching the field on a touch sensitive screen displays the Virtual Keyboard. The Virtual Keyboard is subject to the same security constraints as other input methods.

Accounts

Using the **Options»Accounts…** menu command, anyone with a security level of 10 can create, revise, or delete system user accounts through the **Account Maintenance** dialog box (see Figure 10-1).

Note ***You should carefully consider users and the security level you assign each. Assign only those people responsible for system security level 10 access. Users with security levels of 8 and higher can close process files, exit Lookout, and, in development versions of Lookout, edit process files.***

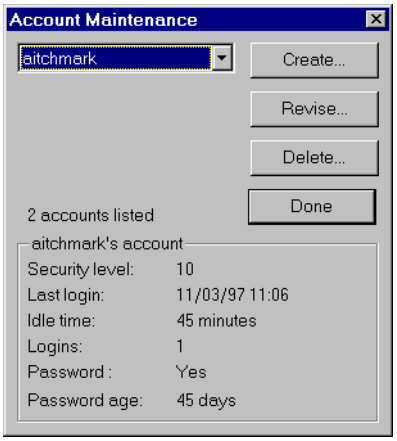
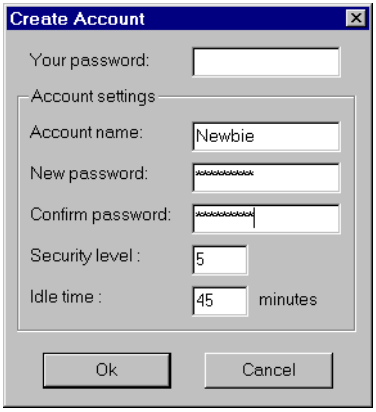


Figure 10-1. Create, Revise, or Delete System User Accounts in the **Account Maintenance** Dialog Box

To delete an account, select the account name and click on the **Delete...** button. To revise an account, select the account name and click on the **Revise...** button. To create an account, click on the **Create...** button. Whether creating or revising an account, you access the following dialog box.



Your password—Enter your password. Lookout requires it before you can add or revise an account.

Account name, **New password** and **Confirm password**—Use any combination of alphanumeric characters. **Security Level** ranges from 1 to 10.

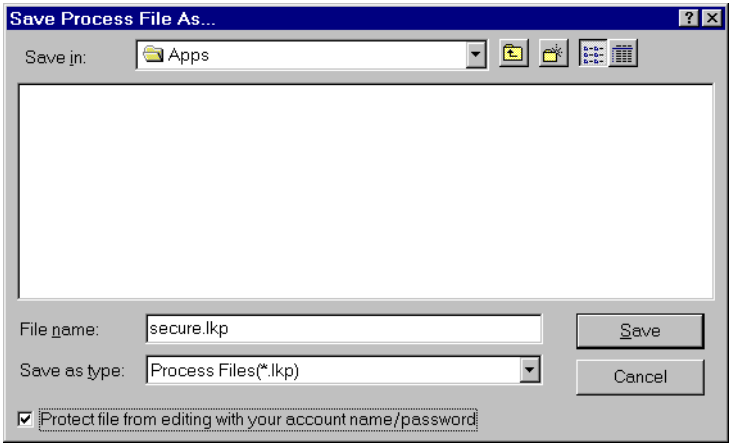
Idle time—The amount of time the computer sits idle (no mouse movement or keyboard action) before Lookout automatically logs off the operator. If you enter `0` (zero), idle time is disabled. For security reasons, you might want to use this feature to automatically log off high level accounts if the computer is left unattended too long. After an account logs off, the account *(nobody)* is logged on.

As with any other account, you can modify the (nobody) account security level. You can set up (nobody) with a security level of 10. If you do set the security level to 10, realize that a low level operator can log off, let (nobody) log on with a security level of 10, and access all Lookout commands. National Instruments recommends that you create your security level 10 accounts and change the (nobody) account to security level 0, which is the only account name that supports security level 0.

☞ **Note** ***If you are the only account with level 10 security and you forget your password, there is no way to access the System»Accounts… command, and there is no way to modify account settings. Contact National Instruments for assistance.***

Process File Security

You can protect your process files from being edited by another person. Log in with an account name and a non-empty password, and select **File»Save As** from the menu. The following dialog box appears.



Check the **Protect file from editing with your account name/password** box at the bottom of the dialog box to save the file with your password as protection. To edit the file again, you have to log in under the same account with the same password.

☞ **Note** ***You cannot open an encrypted file with an earlier version of Lookout, even if you create an account with the same account name and password in that version.***

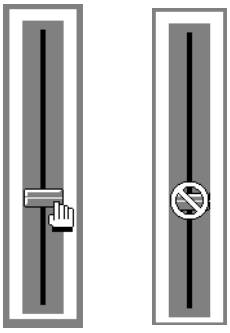
When you protect a process file, Lookout does not save the .LKS file. Because the .LKS file serves as a backup file during application development, you should not use the encrypted-save feature until after you have completed your application and made a backup copy of both the .LKP and .LKS files on a separate archive disk.

Control Security

Several object classes in Lookout support control security, including the Pot, Switch, and Pushbutton. Each class provides some type of control—pots control numeric output signals, and switches and pushbuttons control logical output signals. Each class accepts the **control security level** parameter, which determines whether an operator can control the object. See Chapter 18, *Object Class Reference*, for additional information on these object classes.

To determine if an operator can control a particular object, Lookout compares the security level control of an object to the security level of the currently logged-on account (the operator). If the account security level is equal to or greater than the security level of the object, the mouse cursor

changes into a hand when positioned over the object and the operator can adjust and control of the object.



If the account security level is lower than that of the object, the cursor changes into the international symbol for forbidden, and the operator cannot control the object.

You can implement this feature on an object-by-object basis, instead of for an entire class. System integrators can secure high priority switches, pots, and pushbuttons from operators, while still allowing operators to adjust lower-level security objects.

Lookout globally applies the Control Panel object security setting to all individual objects on that panel and assigns the higher security level (either the control panel or the individual object) when determining whether an operator can control an object. See the discussion of the *Panel* object in Chapter 18, *Object Class Reference*, for additional information.

Viewing Security

Lookout provides viewing security for control panels, controllable objects, and system settings. With these security options, you can restrict access to control panels, objects, and Windows system resources.

Control Panels

A Control Panel object defines viewing security for the entire control panel. For example, if you set **Viewing** security to level 6 on a particular panel, operators with level 5 or lower cannot view that control panel and might not even know that panel exists. If a level 6 (or higher) operator logs on, the control panel instantly becomes available for display. This feature is useful for hiding panels that are rarely used or that contain sensitive information.

Controllable Objects

Controllable objects such as Pots, Switches, and Pushbuttons all have a writable data member called `visible`. When `visible` is true, you can see the object on a control panel. When `visible` is false, you cannot see or adjust the object. To ensure that the object is always visible when it is first created, `visible` defaults to true.

You can connect the `visible` data member of a controllable object (for example, a Pot object) to a controller mode indicator. When the controller is in computer control mode, the `visible` data member of the Pot might be true, allowing the operator to see the Pot and adjust the setpoint. But when the controller is not in computer control mode, the `visible` data member might be false, hiding the Pot from the operator and prohibiting operator control.

You can also use the `username` or `seclevel` data members of the $System object to control the visibility of a control object, depending on the name or security of the person logged on to Lookout at any given time.

System Settings

With the **Options»System…** menu command, you can define system options in the **System Options** dialog box (Figure 10-2) to keep Lookout maximized, the menu bar invisible, title bars invisible, and popups to a minimum.

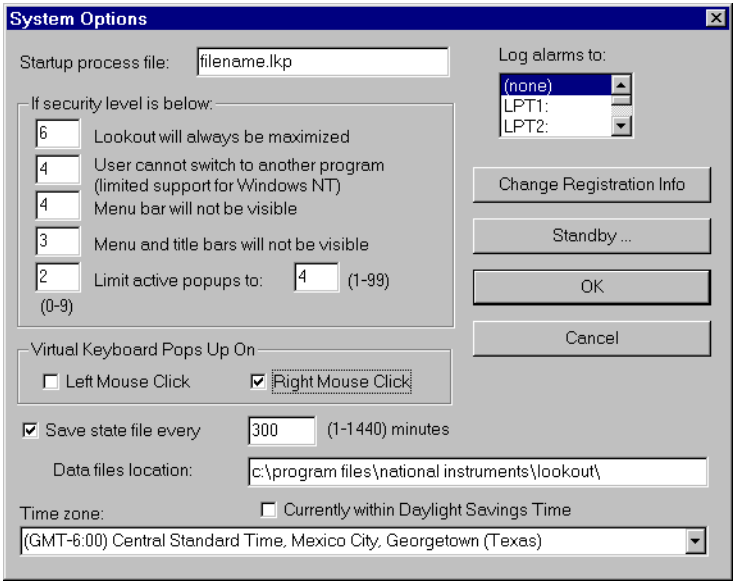


Figure 10-2. Defining Security in the **System Options** Dialog Box

Lookout will always be maximized—When you enter a security level, Lookout prohibits users below that security level from closing Lookout.

Menu bar (and title bars) will not be visible—When you enter a security level, users below that security level cannot view the menu bar or the title bar and, therefore, cannot change to a different Windows application. This feature is not completely supported under Windows NT 4.0. With Windows NT 4.0, you can still use <Ctrl-Esc> or the Windows key to activate the Windows **Start** menu or <Ctrl-Alt-Delete> to bring up the Task Manager.

Limit active popups to:—This option requires two values: a security level and the number of popups. Users below that security level can view up to the specified number of popups at one time. This feature keeps new users from becoming lost. See Options»System menu command in Chapter 16, *Runtime Menu Commands*, for more information.

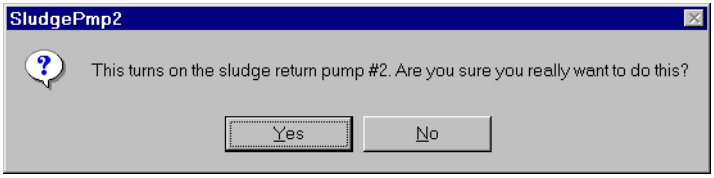
Action Verification

The Switch and Pushbutton object classes support action verification. When you define action verification for an object, Lookout displays a message box stating your **Verify** message and prompts you to select either **Yes** or **Cancel**. If you click on **Yes**, Lookout completes the previous operator command (for example, flips the switch or presses the pushbutton). If you **Cancel**, Lookout ignores the previous operator command.

All action verification parameters accept text expressions, which can contain dynamic data. As an example, consider a switch that controls a pump responsible for filling a storage tank. However, that pump should not fill the tank if the water level is too high. You might enter an expression similar to the following for the Switch **Verify On** parameter:

```
"This turns on the sludge return pump #2. Are you sure
you really want to do this?"
```

The warning message appears every time you turn on the switch. Notice the water level is dynamic—it changes to reflect the value of `level` when the switch is flipped.



When you turn off the switch, no warning message appears because the **Verify Off** parameter was not specified. If you want to disable the **Verify On** warning message, delete the entire expression from the data field.

☞ **Note** ***Pushbutton verification works in much the same way. However, when you select* Yes*, the pushbutton creates only a momentary output signal. When action verification is enabled, it is impossible to hold the button down for any length of time.***

11

Logging Data and Events

This chapter describes three Lookout methods for logging real-time system data to disk—Spreadsheet Logger, Citadel Historical Database Logger, and Event Logger—and report generation.

The **Spreadsheet Logger** creates standard ASCII text files in comma separated value (`.CSV`) file format. You can open and edit these files with common spreadsheet and database programs. The **Citadel Historical Database Logger** creates an historical database that Lookout HyperTrend objects access in real time. You can retrieve this data using Structured Query Language (SQL). The **Event Logger** creates a chronological audit trail of who did what and when.

You might implement all three methods on a single system. They are *not* mutually exclusive.

☞ **Note** ***See Chapter 9,*** *Alarms****, for more information on the Alarm Logger, which logs alarms to disk.***

Spreadsheet Logger

When you want to create permanent ASCII files that you can later open with software packages such as Excel, Lotus, and Foxpro, use a Spreadsheet object to store real-time system data to disk.

You can generate more than one spreadsheet for a single process if you use multiple Spreadsheet objects in one process file (an `.LKP` file). You can have different spreadsheet files in different subdirectories and each might have different data stored using different logging criteria. See the *Spreadsheet Logger* section in Chapter 18, *Object Class Reference*, for more information on the Spreadsheet object.

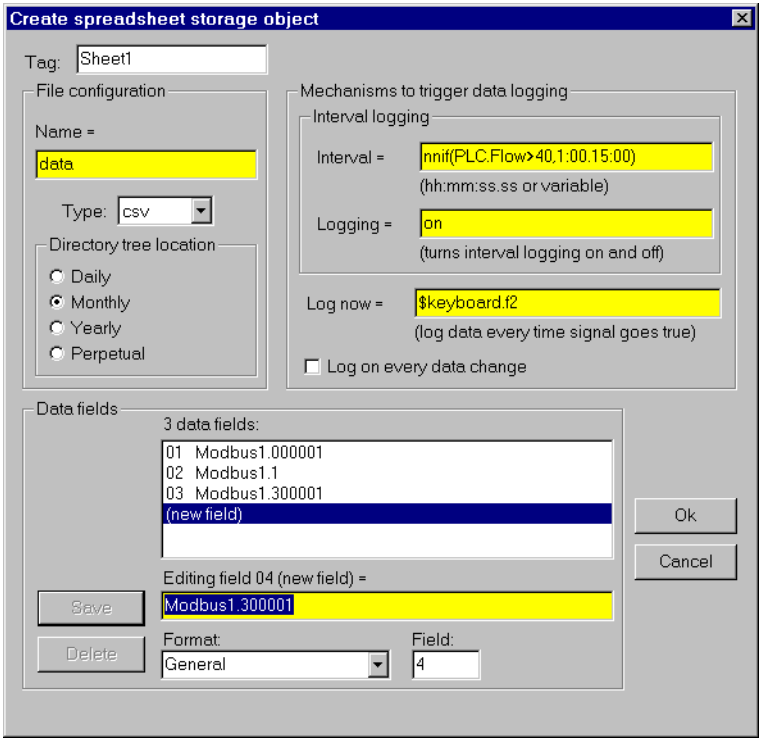


Figure 11-1. A Spreadsheet Object with Complex Mechanisms Defined for Triggering Logging

Data Location

Lookout stores spreadsheet files in subdirectories specified by the **Data files location** field in the **System Options** dialog box (see Figure 11-2).

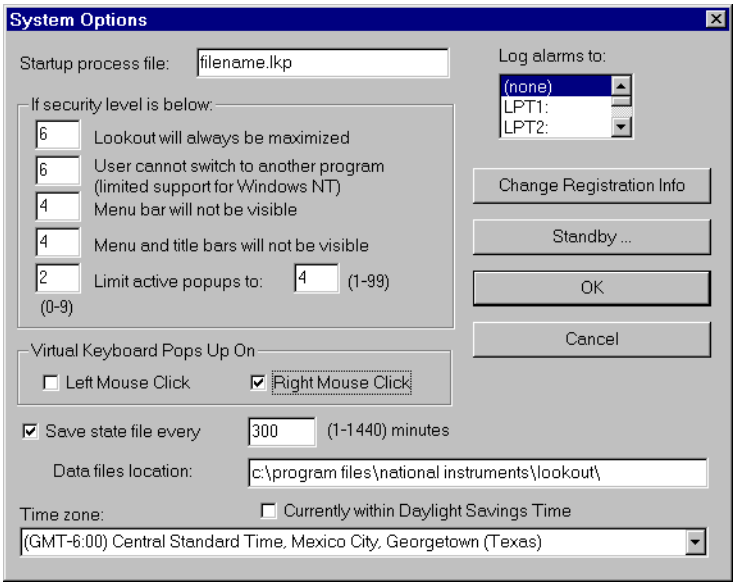
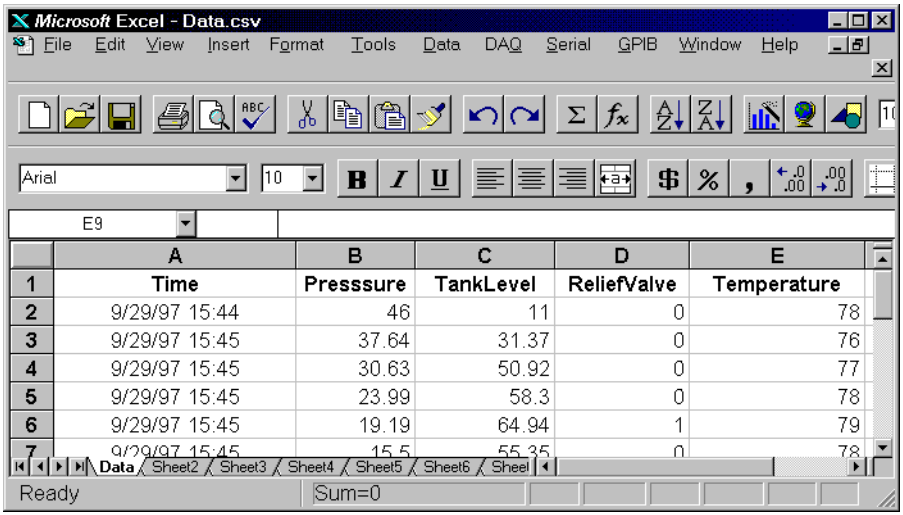


Figure 11-2. System Options Dialog Box Specifies Spreadsheet File Root Directory Path

CSV Files

All Lookout spreadsheet files use the comma-separated value (`.CSV`) format. A `.CSV` file contains text separated by commas. Each line of the file represents a row of spreadsheet data. The first row lists signal names and describes the data stored in each column of the file. The first column of the spreadsheet file specifies the date and time data in each row was logged.



	A	B	C	D	E
1	Time	Presssure	TankLevel	ReliefValve	Temperature
2	9/29/97 15:44	46	11	0	78
3	9/29/97 15:45	37.64	31.37	0	76
4	9/29/97 15:45	30.63	50.92	0	77
5	9/29/97 15:45	23.99	58.3	0	78
6	9/29/97 15:45	19.19	64.94	1	79
7	9/29/97 15:45	15.5	55.35	0	78

Data / Sheet2 / Sheet3 / Sheet4 / Sheet5 / Sheet6 / Sheet

You can view and edit spreadsheet files using a standard word, but spreadsheet programs such as Microsoft Excel provide a more intuitive interface for viewing or editing data and printing custom reports.

File and Disk Errors

If Lookout cannot find the data directory or spreadsheet files or cannot log data to disk, it generates a priority eight alarm with a descriptive error message. For example, you can set the directory tree location to **Monthly** and your **Data files location** as `c:\lookout`. If Lookout encounters a full disk while writing to a file named `data.CSV`, the alarm text reads:

`(Disk is full) “c:\lookout\1994\sep\data.csv”.`

When you correct the problem that caused the alarm, the alarm automatically resets.

Concurrent File Access

When logging data, Lookout opens the file long enough to add a new row of data and then closes the file. Lookout can log approximately 10 new rows of data per second. (The time stamps associated with each row are rounded to the nearest second.) Because Lookout runs under the Windows multitasking environment, you can open the spreadsheet or database while Lookout is running. If you have the data file open in a spreadsheet program such as Excel, Lookout cannot add new rows of data to the file. If Lookout

does not have write access to the file, it generates an alarm explaining the problem:

```
Error writing spreadsheet file: Permission denied
[c:\lookout\1993\sep\data.csv]
```

If Lookout cannot log data to the spreadsheet file, it creates a temporary buffer in memory to which it logs all the spreadsheet data. When you close the spreadsheet file, Lookout updates it and resets the alarm.

Information Overload

If you create a Spreadsheet object that logs fifty data fields (tagnames and/or expressions) at five-minute intervals, you generate 14,400 data points per day or 432,000 data points per month. It can be difficult to determine which data are critical information.

Note ***Carefully consider the importance of each data point and the time interval between data points to avoid information overload for both you and your spreadsheet program.***

If you have logged more than 16,384 rows of data, the current version of Excel cannot load the entire spreadsheet file. Spreadsheets created with one-minute intervals generate 44,641 rows of data if the directory tree location is set to **Monthly**. Spreadsheets created with two-minute intervals generate 22,321 rows of data if the directory tree location is set to **Monthly**. To decrease the number of data rows, try changing the directory tree location to **Daily**, which generates only 1,440 rows of data per day at one-minute logging intervals. You can also use a text editor to divide the larger file into smaller files with fewer data rows.

Besides limiting rows to 16,384, Excel limits spreadsheets to 256 columns of data. One column is reserved for the date and time.

Citadel Historical Database Logger

When you need to view historical information using Lookout HyperTrends or access historical data using Structured Query Language (SQL), use the Citadel Historical Database Logger to store real-time numeric and logical data in a compressed format.

Lookout logs data only when the value of a point changes, *not* at a timed interval. The logged data accurately reflects the actual behavior of the point being logged because the same value of a point is not recorded over and

over. With Citadel, disk usage is exactly proportional to the amount of information recorded.

Once the data is stored to disk, you can view it with a HyperTrend object in Lookout or with SQL. HyperTrends serve as windows into your database. SQL, an industry-standard language supported by most database packages, enables other applications to directly retrieve data from Citadel.

Because Citadel uses Universal Coordinated Time (GMT) to time stamp the data, you do not need to compensate for data logged in different time zones. The HyperTrend object automatically converts from universal to local time before displaying a data history. Identify your time zone in the **System Options** dialog box, shown in Figure 11-2.

Data Location

The **Data files location** field in the **System Options** dialog box specifies the root directory under which all data logging is stored, including the database. Citadel creates a special `CITADEL` subdirectory in the Lookout directory. This subdirectory holds your historical database files. Never move or rename the subdirectory or files.

Unlike spreadsheet files, only one historical database per copy of Lookout exists. If you want to develop multiple process files where each process file has its own unique historical database on a single computer, install Lookout multiple times on the same computer *but in different directories*. You can then specify different locations for each database—one for each copy of Lookout.

Creating a Historical Database

Citadel stores historical information using *traces*. Trace refers to the line of continuity for a specific data member name or point. A trace connects all the historical values for a given point, which displays as a continuous line in a HyperTrend object. If Lookout is unexpectedly interrupted or a data member is temporarily modified to *not* log to disk, a trace can be broken. If the trace is broken, the HyperTrend plots the trace as a continuous line with void sections to represent gaps in the data. You cannot remove individual traces from the database, but you can add new traces to the database by configuring a new point.

You can log any numeric or logical data member of any object to the historical database. Use the following procedure to add a trace to the Citadel database:

1. Select the **Object»Edit Database…** menu command and an object that contains data you want to log.
2. Identify the value you want to log in the **Member** data field. If the value was previously configured, select it from the **Configured points** list.
3. Check the **Log to historical database** checkbox and choose an appropriate **Lifespan**. See *Logging Criteria* for an explanation of time span settings.
4. Enter a value in the **Deviation** field. See *Logging Criteria* for an explanation of **Deviation** settings.

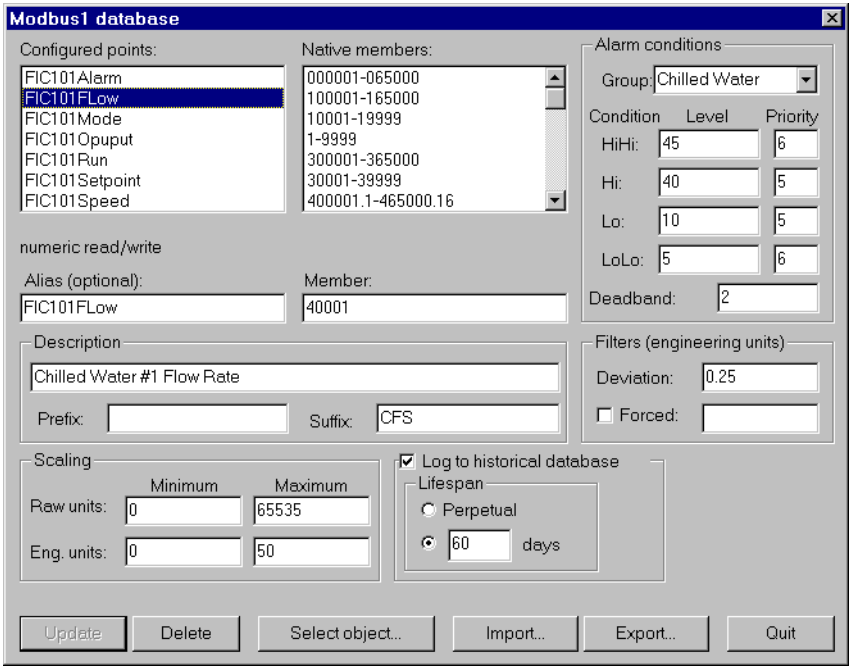


Figure 11-3. Using the **PLC Database** Dialog Box to Create a New Trace

5. Click on the **Save** or **Update** button. (If you are modifying an existing configured point, the button automatically changes to **Update**.)
6. Configure all the object data members that you want logged to the Citadel database.

7. Click on the **Select object** button and select the next object containing data you want to log.
8. Repeat steps 2 through 6 until you have created the traces that you need.
9. Click on **Quit**.

Logging Criteria

To create a new trace in the Citadel historical database, Lookout needs the following information: data member name (**Alias**), **Deviation**, and **Lifespan**.

- Data member name (**Alias**)—Lookout must give a name to each trace in the historical database for archiving and retrieval purposes. If **Alias** is not specified, Lookout uses the native data member name as displayed in the **Member** field. If an alias is assigned to the native data member, Lookout uses the **Alias** name instead. The name used for the trace is the same name saved to the **Configured points** list box. In Figure 11-3, the name is `PLC.FIC101FLOW`.
- **Deviation**—If an analog value surpasses or equals the **Deviation** setting, Lookout saves a new value to disk in the database. If you implement **Scaling** parameters, Lookout compares the **Deviation** setting to changes in the **Eng. units** values to determine when to log a new point. If you leave the **Scaling** parameters blank, Lookout compares the **Deviation** setting to the raw (unscaled) signal. The **Deviation** parameter is not available on logical signals because Lookout logs all state transitions of logical values to disk.
- **Lifespan**—Lookout needs to know how long you want to maintain this trace of data in the historical database. Lookout maintains the data for *at least* the time span specified in the **Log to historical database** parameters.

  Lookout does not discard old data immediately. Disk space containing old data is reused when new data is logged.

Information Overload

Because Lookout creates and maintains the historical database in 1 MB chunks, you must have a least 1 MB of available disk space to begin logging data. When the first 1 MB file fills up with data, Lookout creates a second 1 MB file in the same directory. When the second fills up, Lookout continues to create 1 MB files until all traces can be maintained on disk for the time spans specified in the **Log to historical database lifespan** data

field. Lookout monitors the status of your hard drive and generates an alarm when your disk falls below 500 KB of available space.

Use discretion when determining **Deviation** settings on each trace of data. If **Deviation** is not specified, *any* change of an analog value results in a new point being logged to disk, and you might generate too much data. If the **Deviation** setting is too large, very little information might be saved to disk.

Event Logger

When you want to create a chronological schedule of events or an audit trail, use the Lookout Event Logger. The Event Logger logs all events: operator commands, Event objects, and all global events (running Lookout, closing Lookout, opening a process file, closing a process file, entering edit mode, exiting edit mode, and logging on and off).

Operator commands include events such as closing a process file, entering edit mode, and adjusting a pot. Along with each event, Lookout logs the account name (the operator), date and time of the event, tagname of the object adjusted, and the previous and subsequent settings of the object. Because all of this information is logged, you create an exhaustive audit trail for that specific workstation.

Use the Event object class to define and log your own event messages based on a user-defined trigger. See the *Event* section in Chapter 18, *Object Class Reference*, for additional information.

Data Location

Lookout stores event files named `EVENT.DAT` in monthly subdirectories specified by the **Data files location** parameter in the **System Options** dialog box. At the beginning of each month, Lookout creates a new monthly subdirectory and event file. If you specify `C:\DATA` as the **Data files location**, Lookout saves your event files as follows:

```
C:\DATA\1993\OCT\EVENT.DAT
            \NOV\EVENT.DAT
            \DEC\EVENT.DAT
       \1994\JAN\EVENT.DAT
            \FEB\EVENT.DAT
```

To view these files, print them out with the **File»Print»Events…** command.

Information Overload

The event logger can store information on an individual object instance for switches, pots, and pushbuttons. Every time you flip a switch, adjust a pot, or depress a pushbutton, Lookout logs the event to disk. If you enable the Event Logger for all instances of these object classes, you might collect more data than you need.

By selectively enabling the Event Logger on critical switches, pots, and pushbuttons, you cut down on the amount of information logged to the event file. For example, you might place a dozen switches on a control panel, but only one of those twelve requires data logging. Rather than log events for all twelve switches, enable the Event Logger only on that single switch.

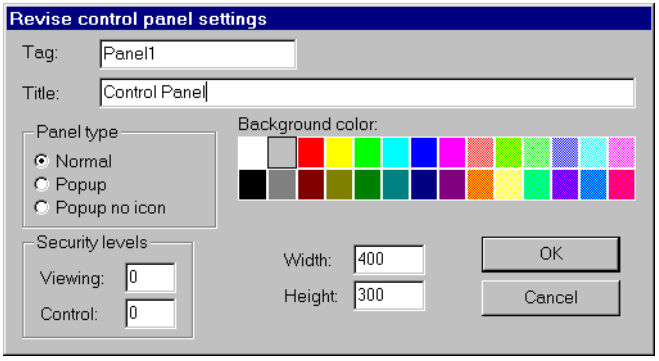
Report Generation

Control Panel Reports

Panel objects have a special data member named `print`. When `print` changes from FALSE to TRUE, Lookout copies a picture of the control panel and sends that copy to your printer using the Windows Print Manager. See *Panel* section in Chapter 18, *Object Class Reference*.

To configure your custom report, design a control panel and connect a logical trigger signal to the panel. You can connect any logical expression to the `print` member. The control panel does not have to be visible to be printed. Therefore, an operator can view one screen (panel) while printing another screen.

Lookout determines the printed panel size based on dimensions entered in the **Revise control panel settings** dialog box.



When **Panel type** is set to **Normal**, Lookout assumes that the panel is maximized. Even though your control panel might be maximized, Lookout prints only the area defined by your Panel object **Width** and **Height** settings. For example, you might have a panel that looks like Figure 11-4 when maximized.

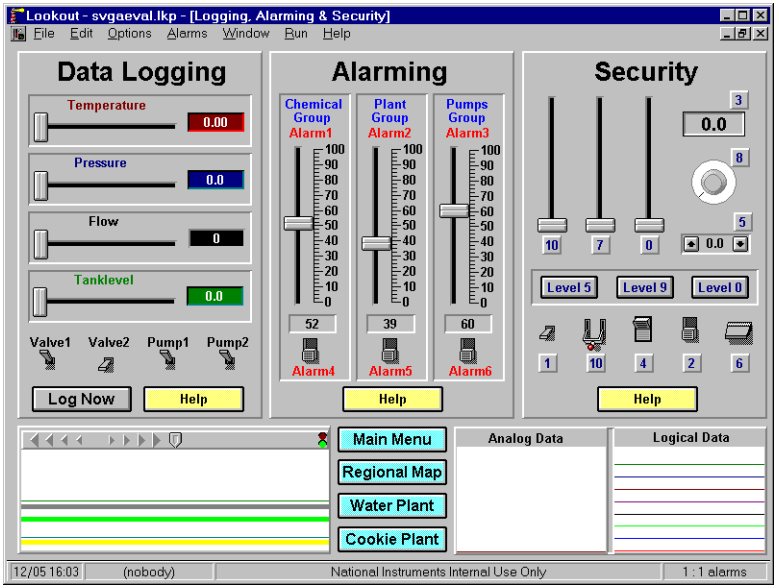
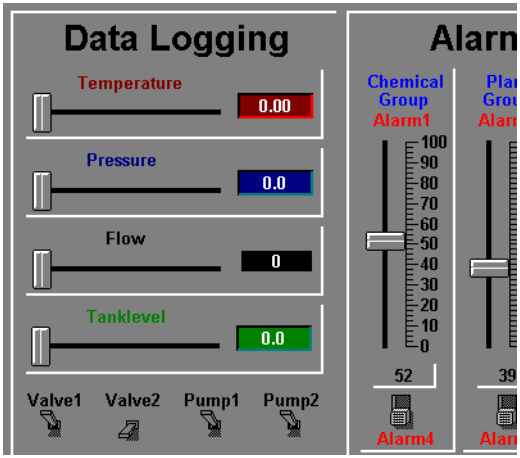


Figure 11-4. Printing a Normal Panel Object

The panel in Figure 11-4 has a specified **Height** of 300 pixels and a **Width** of 400 pixels. At a video resolution of 800 × 600, the printed panel would look something like the following figure.



Modify the **Height** and **Width** dimensions of your Panel object to match your screen resolution. This problem applies only to **Normal** style panels. **Popup** and **Popup no icon** style panels print according to the WYSIWYG principle (What You See Is What You Get).

Third-Party Reports

You can generate reports using Lookout and a third-party spreadsheet program. Current spreadsheet software provides extensive macro command capabilities you can use to quickly generate reports from Lookout.You can write Excel macros that prompt you for the report date, automatically open the correct spreadsheet file(s), extract the data, and print reports with combinations of numeric tables and graphs. You can launch these macros from Lookout or automatically invoke them using predefined timers.

12

Structured Query Language

This chapter describes Structured Query Language (SQL), Open Database Connectivity (ODBC), and accessing Citadel data using both SQL and ODBC.

☞ **Note** ***The Express version of Lookout does not log data or contain tools for extracting logged data.***

Introduction

The Citadel historical database includes an Open Database Connectivity (ODBC) driver, which enables other applications to directly retrieve data from Citadel using Structured Query Language (SQL) queries.

What is ODBC?

ODBC is a standard developed by Microsoft. It defines the mechanisms for accessing data residing in database management systems (DBMSs). Nearly all Windows-based applications that can retrieve data from a database support ODBC.

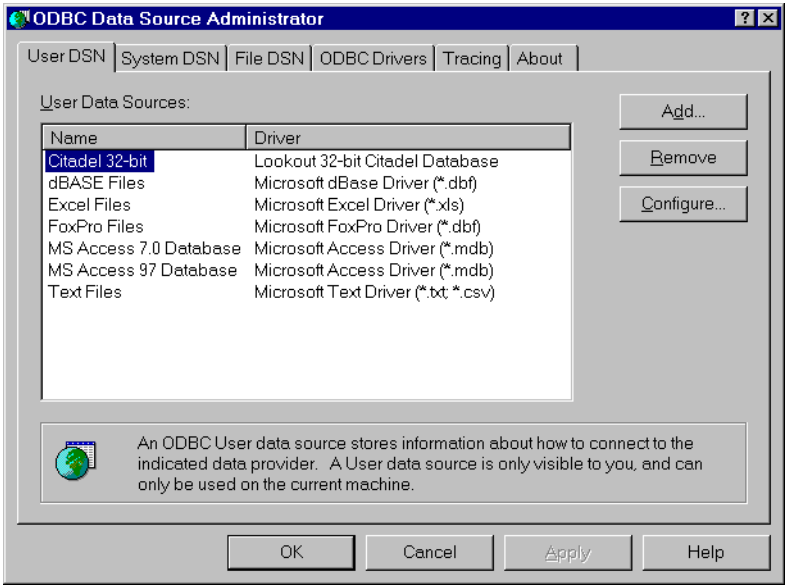
What is SQL?

Structured Query Language (SQL) is an industry-standard language used for retrieving, updating and managing data. In Lookout, you can use SQL to build queries to extract data from Citadel. The Citadel ODBC driver also includes many built-in data transforms to simplify statistical analysis of retrieved data.

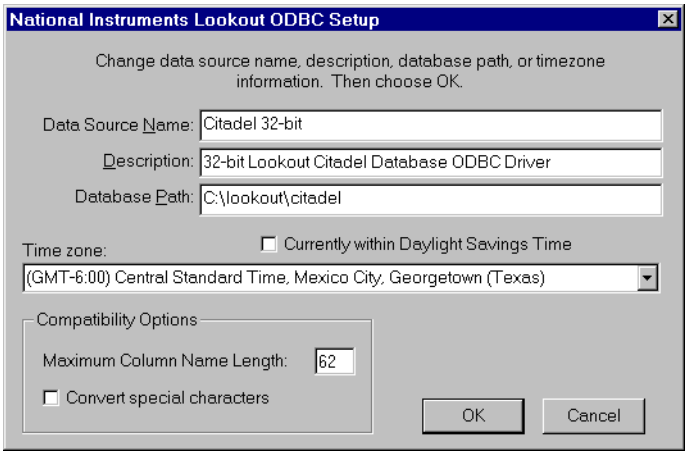
Configuring the Citadel ODBC Driver

1. Shut down all applications that might use ODBC. Applications such as spreadsheets, word processors, database programs, and Microsoft Query might use ODBC. You also need to shut down Lookout if you are using the SQL object.

2. If you are using Windows 95, click the **Start** button and select **Settings»Control Panel**. Otherwise, choose the Control Panel icon from the Main program group.
3. In the **Control Panel** window, choose the 32-bit ODBC icon or the ODBC icon if you are using the 16-bit ODBC driver.
4. In the **Data Sources** dialog box, choose **Drivers...**.



5. Choose the `Citadel` driver and select **Setup...**.



6. Make changes as appropriate. For example, if Lookout and the Citadel database are not in the `C:\LOOKOUT` directory, you might have to modify the database path.

☞ **Note** ***Some applications are not completely ODBC compliant. If you plan to use Microsoft Query, Microsoft Access, or Visual Basic, ensure*** *Maximum Column Name Length* ***does not exceed 62 characters. These applications cannot handle longer tag names. Applications that are completely ODBC-compliant can handle tag names up to 126 characters long. All traces whose tag names exceed the*** *Maximum Column Name Length* ***are excluded from queries.***

☞ **Note** ***If you plan to use Microsoft Access or Visual Basic, select*** *Convert special characters* ***to force Lookout tag names into an accepted format by replacing characters within the tag names with the characters in Table 12-1.***

Table 12-1. Special Access SQL Characters

Special Character	Converted Character
period (.)	backslash (\ )
ampersand (&)	at sign (@)
exclamation (!)	vertical bar (|)

7. Select **OK** and **Close** to exit.

Accessing Citadel Data

Traces Table

The ODBC driver presents Citadel data to other applications as a *traces* table. The table contains a field or column for each data member logged to the Citadel database and three fields you can use to specify query criteria and to time-stamp retrieved data: `Interval`, `LocalTime`, and `UTCTime`.

Interval specifies the query value sample rate. **Interval** can range from 10 ms to several years. **Interval** defaults to 1 (one day).

Because Citadel is event-driven, it only logs a value when the value changes. Using **Interval**, you can query Citadel for values evenly spaced over a period of time.

LocalTime and **UTCTime** are time-stamps that indicate when values are logged. Citadel stores the time in **UTCTime** format and derives **LocalTime** from the stored time.

The following *where clause* query uses **Interval** and **LocalTime** to select data over a specified time at one-minute intervals. Notice that time and date formats are the same as those used in Lookout.

```
SELECT * FROM Traces
WHERE LocalTime>“12/1 10:00”
   AND LocalTime<“12/2 13:00”
   AND Interval=“1:00”
```

Data Transforms

Your queries can include special commands that perform data transforms to manipulate and analyze historical data. See Table 12-2 for a list data transform commands.

Table 12-2. Data Transforms

Command	Transformation
Min{Datapoint}	Returns the minimum for *Datapoint* across the interval.
Max{Datapoint}	Returns the maximum for *Datapoint* across the interval.
Avg{Datapoint}	Returns the average for *Datapoint* across the interval.
Stdev{Datapoint}	Returns the standard deviation for *Datapoint* across the interval.
Starts{Datapoint}	Returns the number of starts (that is, the number of transitions from OFF to ON) for *Datapoint* across the interval. For numeric points, 0.0 is interpreted as OFF, and all other numbers are treated as ON.
Stops{Datapoint}	Returns the number of stops (that is, the number of transitions from ON to OFF) for *Datapoint* across the interval.
ETM{Datapoint}	Returns the amount of time *Datapoint* was in the ON state across the interval.
Qual{Datapoint}	There might be gaps in the historical data traces in Citadel because of machine shutdown, Lookout shutdown, or a similar occurrences. Qual returns the ratio of time for which valid data exists for *Datapoint* across the interval to the length of the interval itself. If valid data exists for only one-half of the interval, Qual returns 0.5.

Using these data transforms you can directly calculate and retrieve complex information from the database such as averages and standard deviations, so you do not need to extract raw data and then manipulate it in another application.

For example, you need to know how many times a compressor motor started in December. You also need to know its total runtime for the month. Use the following query to get your answers:

```
SELECT“Starts{PLC.MotorRun}”,
       “ETM{PLC.MotorRun}”
FROM Traces
WHERE LocalTime>=“12/1/95”
  AND LocalTime<“1/1/96”
  AND Interval=“31”
```

SQL Examples

The following examples are typical query statements; however, your queries might be much more involved, depending on your system requirements.

```
SELECT *
FROM Traces
```

Retrieves the *current* value of every data member logged to Citadel. Because your query does not occur at the same moment in time as a PLC poll, signals scanned from PLCs are not included in the retrieved data.

```
SELECT *
FROM Traces
WHERE Interval=“0:01”
```

Retrieves the value of every data member logged today in one-second increments. Notice that the interval value is enclosed in quotation marks.

```
SELECT LocalTime, “Pot1”
FROM Traces
WHERE LocalTime>“8:50”
  AND Interval=“0:01”
```

Retrieves and time-stamps the value of Pot1 in one-second increments from 8:50 this morning to now. Tag names are enclosed by quotes.

```
SELECT LocalTime, “AB1.I:3”, “Max{AB1.I:3}”
FROM Traces
WHERE LocalTime>“10/1/95”
  AND LocalTime<“11/1/95”
  AND Interval=“1:00”
```

Retrieves and time-stamps an Allen-Bradley PLC input in one-minute intervals for the month of October. This query also indicates the highest occurring input value of each minute.

```
SELECT LocalTime, “OVEN1_SP”, “PLC.OVEN1_PV”,
     “Max{PLC.OVEN1_PV}”, “Min{PLC.OVEN1_PV}”,
     “Avg{PLC.OVEN1_PV}”
FROM Traces
WHERE LocalTime>=“14:00”
   AND LocalTime <“15:00”
   AND Interval=“1:00:00”
```

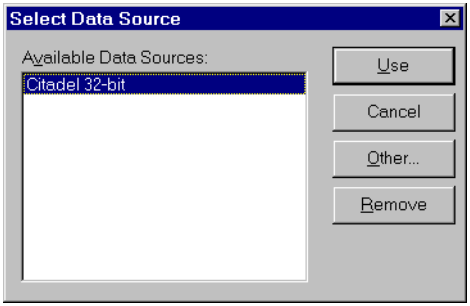
Retrieves an oven temperature at 3:00 p.m. and shows the highest, lowest, and average temperatures between 2 p.m. and 3 p.m.

Accessing Citadel Data with Microsoft Query

Microsoft Query is a graphical data retrieval tool supplied with Microsoft Office and Microsoft Excel that you can use to build your SQL statement using interactive dialog boxes.

The following example uses Microsoft Query Version 2. If you have a different version, some of the steps or dialog boxes may vary. Refer to your Microsoft documentation for further information.

1. Launch Microsoft Query.
2. Choose **File»New…** and select the `Citadel` data source.



☞ **Note** ***If Microsoft Query is unable to connect to Citadel, make sure you have logged data to Citadel and entered the correct*** *Database path* ***in the*** *ODBC Setup* ***dialog box.***

Note ***If Citadel is not listed in the*** *Data Source* ***dialog box, it might not have been accessed yet. Choose*** *Browse…* ***and select*** `Citadel` ***from the ODBC data sources. If Citadel is not listed as an ODBC Data Source, you need to install it. Citadel is installed as a part of the Lookout installation process. If you did not install Citadel when you installed Lookout, you should re-install the complete package.***

3. In the **Add Tables** dialog box, double click `Traces`.

4. Close the dialog box.

   Microsoft Query presents the full Query Window with the `Traces` table. The tag names in the `Traces` table is a comprehensive list of all tag values that have been logged to Citadel.

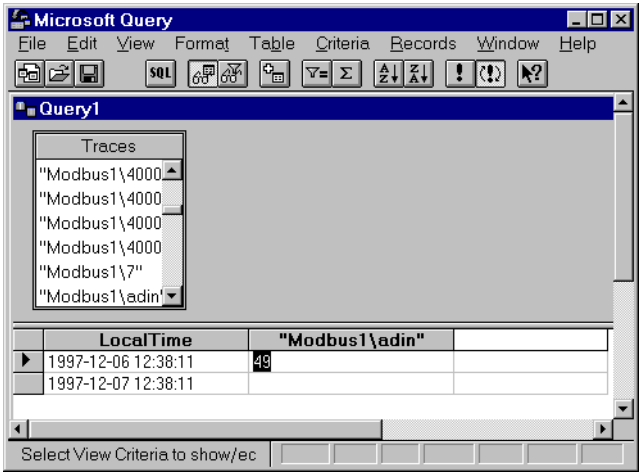


Figure 12-1. The Query Window with `Traces` Table

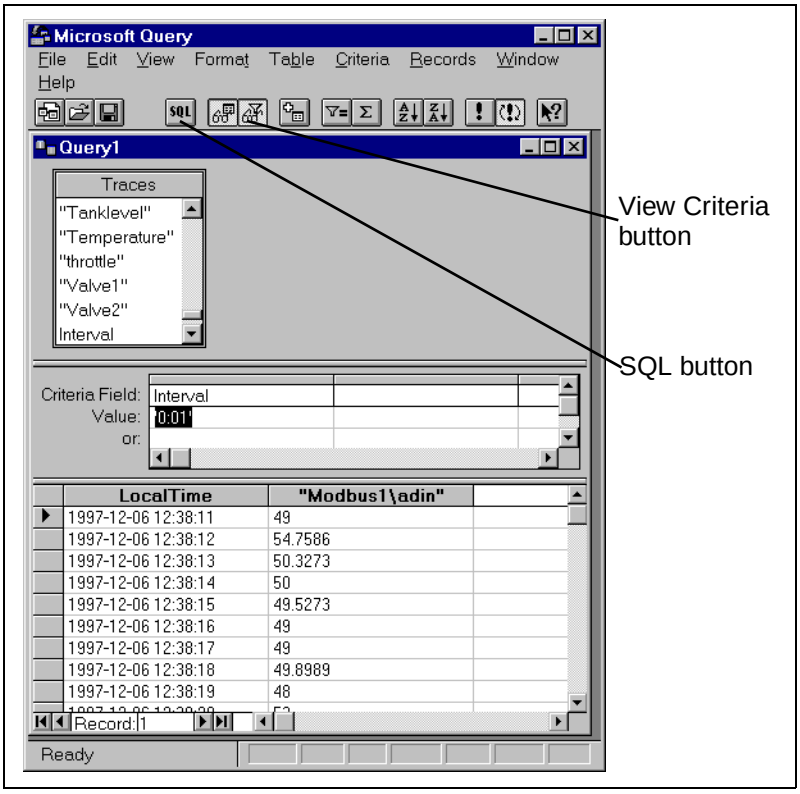
5. To view a trace, double click on or drag the field you want to view to a blank column in the data pane. In the proceeding illustration, LocalTime and "Modbus1.adin" have been dragged down.

6. To view a data transform value, enter the function directly into a blank column. For example, to view the minimum value of PLC.AI2, you would enter `"min{PLC.AI2}"`. You must include the quotation marks and braces.

   The data set in Figure 12-1 was retrieved using no specific criteria, so the ODBC driver used the default. Although there are several ways to specify criteria, in this example, we use the criteria pane.

7. Click on the **View Criteria** button. The pane appears in the Query window.

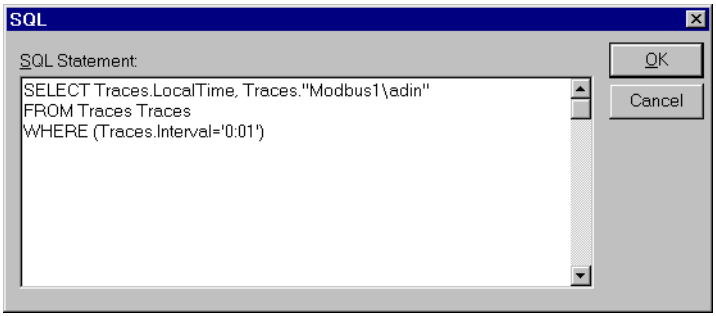
8. Add a field to the criteria pane by double-clicking on the field, or by dragging it to the blank column in the criteria pane.



When you enter qualifying criteria values, use the syntax demonstrated in *SQL Examples* earlier in this chapter. See the following illustration.

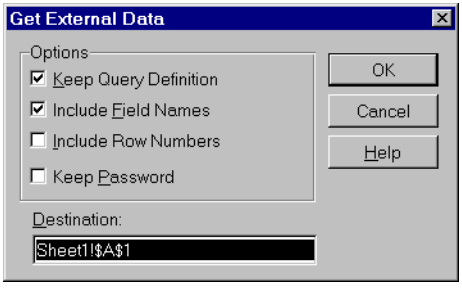
As soon as you specify criteria, Microsoft Query retrieves the specified data. You can save your query at any stage of development, and, as you build your query, the application builds an SQL statement.

9. Click on the **SQL** button to view or edit the query statement.



Accessing Citadel Data with Microsoft Excel

1. Launch Excel.
2. Choose **Data»Get External Data…**. This Excel command directly launches Microsoft Query.
3. Use an existing query or create a new one. See *Accessing Citadel Data with Microsoft Query* for information on querying.
4. When you finish building your query, return the result set with the **File»Return Data to Microsoft Excel…** command. Excel prompts you with the **Get External Data** dialog box, enabling you to change or confirm the destination cells of the result set. If you want to query Citadel later to update the result set, check the **Keep Query Definition** checkbox.
5. Click on **OK** to write the data to an Excel worksheet.



6. To update your result set, select any cell within the worksheet result set, choose **Data»Get External Data…** command, and click on the **Refresh** button.

Accessing Citadel Data with Microsoft Access

☞ **Note** ***The SQL/92 standard states that a delimited identifier is any string of no more than 128 characters enclosed in quotation marks. It further states that characters within a delimited identifier are exempt from SQL syntax checking. Unfortunately, Microsoft Access performs its own syntax checking for ODBC queries using a non-standard SQL syntax—even within delimited identifiers. For this reason, National Instruments provides a*** *Convert Special Characters* ***selection in the*** *Citadel ODBC Setup* ***dialog box. When selected, the ODBC driver converts the special characters to accepted Access characters. See Table 12-1 for a complete list of special characters.***

When you query Citadel using Microsoft Access, you must use the Microsoft Access non-standard SQL syntax in your select statement. Remember to consider the following elements:

- Convert special characters for Access compatibility (see Table 12-1)
- Place double quotes around Lookout trace names to identify them as delimited identifiers
- Enclose identifiers in square brackets ([ ])
- Place pound signs (#) around time stamps

For example, to retrieve `LocalTime`, `Pot1`, and `Tiway1.V101` where `LocalTime` is greater than 11/20/95 18:00:00, and where `Interval` is one second, enter the following query:

```
SELECT LocalTime, ["Pot1"], ["Tiway1\V101"]
FROM Traces
WHERE LocalTime > #11/20/95 6:00:00 PM#
AND Interval = '0:01'
```

1. To query Citadel from within Microsoft Access, open a database and use the **File»Get External Data»Link Table** command.
2. Choose the **Machine Data Source** tab in the **Select Data Source** dialog box.

3. Highlight `Citadel` as shown in following illustration.

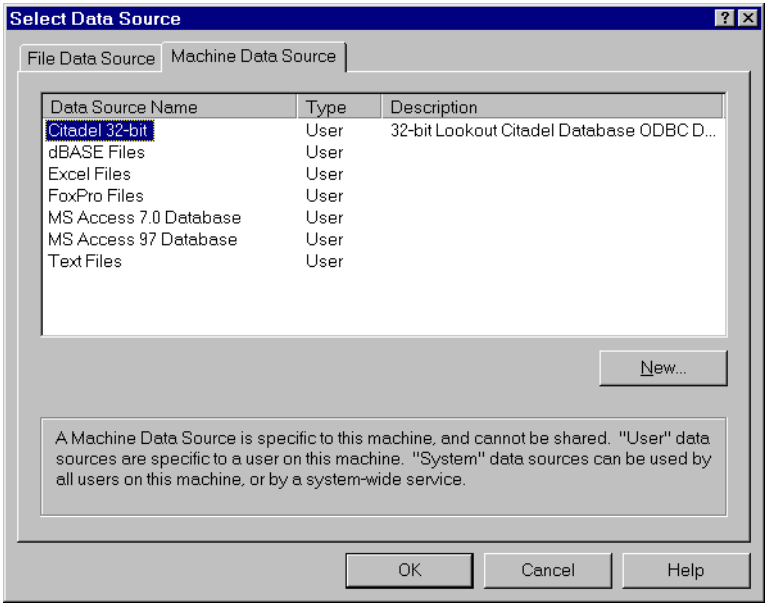
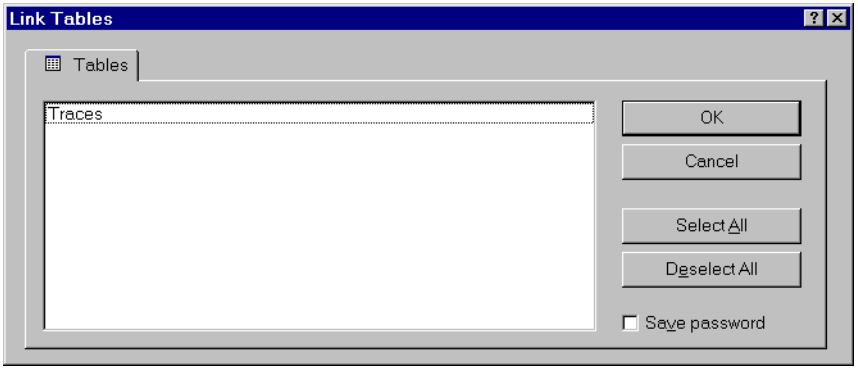


Figure 12-2. The **Select Data Source** Dialog Box

4. Click on **OK**.
5. In the **Link Tables** dialog box, choose `Traces`. The new table links to your database.



6. At this point, you can build queries in Access to extract data directly from the Citadel database. See your Access documentation for more detailed information on this process.

Accessing Citadel Data with Visual Basic

☞ **Note** ***Visual Basic software relies on Microsoft Access DLLs for performing ODBC queries. Because it uses the Access non-standard SQL syntax, be sure that*** *Convert Special Characters* ***is selected in the Citadel ODBC Setup dialog box. See the note under*** *Accessing Citadel Data with Microsoft Access.*

You use the Citadel ODBC Driver in Visual Basic the same way you would use any other ODBC driver. To retrieve and view data, you have to create a data control and at least one text control.

1. Place a data control on an open form.
2. Set its **Connect** property to `DSN=Citadel` and double-click on its **Record Source** property to identify `Traces` as its source table.
3. If you want to select all of the data for all traces in the Citadel database, leave the **Record Source** property set to `Traces`. If you want to narrow your query, enter an SQL select statement in the **Record Source** property.

   For example, to retrieve `LocalTime`, `Pot1`, and `AB1.I:3` where `LocalTime` is greater than 11/20/95 18:00:00 and less than 18:30:00, and where `Interval` is one minute, enter the following query:

```
SELECT LocalTime, ["Pot1"], ["AB1\I:3"]
FROM Traces
WHERE LocalTime > #11/20/95 6:00:00 PM#
AND LocalTime < #11/20/95 6:30:00 PM#
AND Interval = '1:0'
```

☞ **Note** ***Remember to use the Microsoft Access SQL syntax in the select statement, convert special characters for Access compatibility (see Table 12-1), place double quotes around Lookout trace names to identify them as delimited identifiers, enclose identifiers in square brackets (*** **[ ]** ***), and place pound signs (#) around time stamps.***

4. Place a text control on the form.
5. Set its **Data Source** property to the name of your data control—for example, `Data1`.
6. Click on the **Data Field** property to highlight it and then use the property sheet drop-down combo box to select the desired field name. All logged data members should be listed including `LocalTime`, `Interval`, and `Pot1`.
7. Repeat steps 4 through 6 for each data member you want to display on your form.

13

Dynamic Data Exchange

This chapter explains how to use Dynamic Data Exchange (DDE) with Lookout. DDE is the Microsoft message-based protocol used by applications like Microsoft Excel and Lookout to link to data in other applications.

When the data in a source application changes, it dynamically updates all linked data (in real-time). With DDE you can dynamically link other Windows applications to Lookout.

There are several DDE protocol formats. Lookout supports the standard Microsoft formats, XlTable and CF-TEXT. XlTable is often referred to as the Fast table format; CF-TEXT is often called text format. Lookout also supports hot DDE links and NetDDE.

Any two applications participating in dynamic data exchange are engaging in a DDE conversation. In such a conversation, Lookout acts as either the client application or the server application (or both, in a peer-to-peer configuration). If Lookout is getting data from another application, Lookout is the *client*. But if another application is getting data from Lookout, Lookout is the *server*.

The client application is responsible for establishing a DDE link with the server. When Lookout is a client, it first tries to establish an XlTable DDE connection (because this is the most efficient). If the server application does not support this format, Lookout uses the CF_TEXT DDE format.

To establish a DDE link, the client application must identify the location of the desired data. A three-tier address identifies the location of the data: *Service*, *Topic*, and *Item*. Look in the application documentation to determine its service, topic, and item.

Service specifies the name of the server application the client is linking to. Each application that supports DDE has a unique service name. For example, `LOOKOUT` is the service name of Lookout, and `EXCEL` is the service name of Microsoft Excel.

Topic is the second level in the three-tier address. For many server applications like Excel and Lookout, topic specifies a particular file. In

Lookout, the topic is the process file name, minus the `.LKP` extension. For example, you would refer to a process file named `PLANT.LKP` as `Plant` when using it as the topic in a DDE link.

Item identifies the specific data or value being linked between the server and the client. A Lookout item is the object tagname, followed by a specific data member (such as `TagName.datamember`) if needed. See *Identifying Object Data Members* in Chapter 4, *Using Lookout*, for detailed information on selecting objects and data members. An item in a spreadsheet is the name of a cell, such as B3.

Linking Lookout to Other Applications

Lookout can act as a DDE Client, DDE Server, and both DDE Client and Server. Therefore, there are three basic ways to link Lookout to another application using DDE:

- Lookout as the Server
- Lookout as the Client
- Lookout as both Client and Server (peer-to-peer)

☞ **Note** ***All readable numeric, logical, and text values in Lookout are automatically available to any other application through DDE. No special setup is required.***

Because Microsoft Excel is widely used and accepted, it is used in the Lookout DDE examples.

DDE Server

In this example, you can send information from Lookout to another application, making Lookout the server. First, create a potentiometer in Lookout so you can link its value in real-time to a cell in Excel. Any time the pot is adjusted, the value in the spreadsheet cell automatically changes.

1. Ensure Lookout is not in Edit Mode.
2. Hold down the <Ctrl> key and click the left mouse button on the object you want to link to. In this case, select the Pot object you just created.
3. Lookout beeps when it successfully copies the object value to the clipboard. The object can be a slider, bar graph, switch, pushbutton, digital display, text entry object, knob or almost anything else in Lookout that contains a value.
4. Start Excel and select the cell you want to link to.

5. If your version of Excel is 5.0 or above, select **Edit»Paste Special…**. Then click on **Paste Link**. If you have an older version of Excel, select **Edit»Paste Link**.

You have just created your first DDE link. Repeat this process as many times as is needed. If you are linking large numbers of objects to Excel, you might want to use the Excel copy and edit tools to speed up the process.

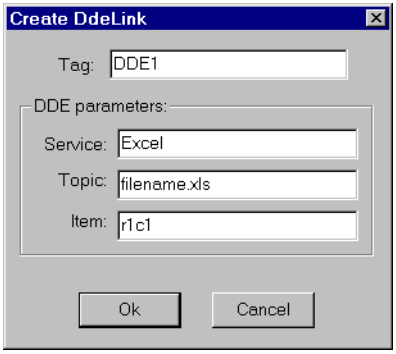
Not all applications support the Windows Clipboard shortcut method as described above. Therefore, you may have to manually enter the appropriate Lookout Service, Topic, and Item in the other application to create a DDE link to that package. The format in which you enter this information varies from one package to another. For this reason, you should refer to the documentation of the client application for instructions.

DDE Client

In this example you import information from another application into Lookout. For instance, you might want to use a value calculated inside a spreadsheet as a process control setpoint for a Lookout application. In this kind of DDE link, Lookout is the client and the spreadsheet application is the server. Because Lookout is the client, it is responsible for establishing the link to the server data. Therefore, you must identify the service, topic and item in Lookout. These are object parameters in the DdeLink object class.

☞ **Note** ***You cannot create or modify objects in Runtime-Only versions of Lookout. Therefore, you must perform this procedure using a Development/Runtime Lookout system.***

1. Select **Object»Create…** and select the DdeLink class.



2. In **Service**, enter the name of the software package (`Excel` in this example).
3. In **Topic**, enter the name of the spreadsheet file.
4. In **Item**, enter the address of the cell you want to read a value from.

   Notice that the entered cell address is `r1c1`. This translates to row1/column1 (cell A1) in Excel. The Excel DDE structure requires this format.
5. Click on **OK**, and then select **OK** again when Lookout prompts you to insert the expression `DDE1`. Finally, pick the desired display format and click on **OK**.

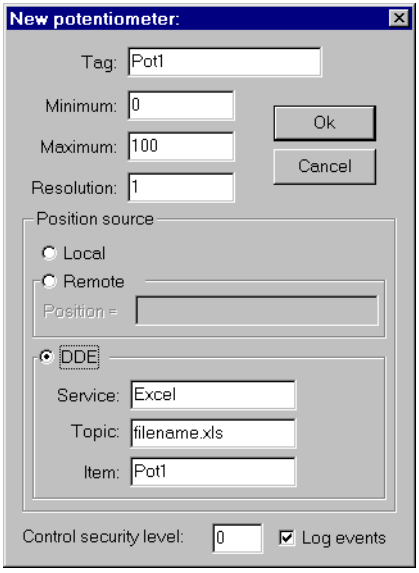
To test your link, enter a numeric value into cell A1 of your spreadsheet. Whatever value you enter into the spreadsheet is immediately written to the DDE expression on your panel. Of course, you can connect the DDE object you just created to other Lookout objects. See *DdeLink* definition in Chapter 18, *Object Class Reference*, for more information.

☞ **Note** ***The DdeTable object is another way of linking data to Lookout using DDE. This object class links large quantities of data though the more efficient XlTable format. See*** *DdeTable* ***definition in Chapter 18,*** *Object Class Reference****, for more information****.*

DDE Peer-to-Peer

Assume you want to take the Lookout as a Server example one step farther. Suppose you want to adjust the pot to change a value in Excel and also be able to enter a different value in Excel to adjust the pot. That is, to send data both ways through a DDE link. You can easily create such two-way links for user-controlled objects (that is, switches, pots and pushbuttons).

☞ **Note** ***You cannot create or modify objects in Runtime-Only versions of Lookout. Therefore, you must perform this procedure using a Development/Runtime Lookout system.***



Select **Object»Create…** and define a new Pot; or select **Object»Modify** and select the existing Pot object.

1. Change **Position source** from Local to DDE.
2. In **Service**, enter the name of the software package (Excel).
3. In **Topic**, enter the name of the data file.
4. In **Item**, enter the address of the cell you want to read a value from.
5. Click on **OK** to create or modify the definition of the object.
6. If the object is new, insert its display member into the panel so you can test your link.

To test your link, enter a value into the spreadsheet cell you specified and watch the pot. Then adjust the pot and watch the spreadsheet cell. You should see the values within the two applications change in unison.

☞ **Note** ***If you link to a Pot object, the linked value is numeric, so you enter a numeric value into the spreadsheet cell. But if you link to a Switch or Pushbutton object, the linked value is logical. Linked logical values are shown in spreadsheet cells as*** `true` ***or*** `false`***. To change the value of a logical value in a spreadsheet cell, enter*** `true` ***or*** `false`**,** `0` ***or*** `1`**,** `on` ***or*** `off`**.**

DDE Alarms

Cannot establish DDE conversation with <service>, <topic>
This alarm occurs if a Lookout client is unable to connect to the server corresponding to the given service and topic. The alarm also occurs if the server terminates the conversation (for example, if the server is shut down). The alarm is deactivated when the Lookout client successfully connects to an item on the server.

Verify that the service and topic were typed correctly when creating the object that is using DDE. Verify that the server application is running. If the server is on another computer on the network, verify that the network is up. If the server is on a computer running Windows NT, verify that you are authorized to log on to that computer and that the current user logged onto the NT machine has trusted the DDE share to which you are trying to connect (see the discussion of trusted DDE shares in the Networking chapter of this manual).

DDE client error for <service>, <topic>, <item>: (received NACK for advise)
DDE client error for <service>, <topic>, <item>: (received NACK for request)
Verify that the named item exists on the server and that the server supports DDE links for the item. This alarm occurs when Lookout is a client.

DDE client error for <service>, <topic>, <item>: (received NACK for poke)
This alarm occurs if you are using Lookout as both client and server, and have made a remote connection to an item that is not writable in a DataTable, Pushbutton, Pot, Switch, or TextEntry. The only Lookout objects that support writes (pokes) are DataTable, PushButton, Pot, Switch, and TextEntry. These support writes into their implicit data members only.

If the server is running Windows NT, it is possible that the DDE share on the computer is configured to support reads (advises) but not writes (pokes).

DDE client error for <service>, <topic>, <item>: (advise timed out)
DDE client error for <service>, <topic>, <item>: (request timed out)
DDE client error for <service>, <topic>, <item>: (poke timed out)
Verify that the server application is running. Verify that the item exists on the server. If the server is on another computer on the network, verify that the network is up.

DDE client error for <service>, <topic>, <item>: (received invalid data)
DDE server: corrupt data block poked to item <item>, topic <topic>
Either the server received a corrupt data block from the client or the client received a corrupt data block from the server. This may be the result of network trouble. If the alarm is consistent and predictable, then you may have discovered a bug. Call National Instruments technical support for further help.

DDE server: failed to post advise for item <item>, topic <topic>
Verify that the client application is running. Verify that the item still exists on the client. If the client is on another computer on the network, verify that the network is up.

14

Networking

With Lookout you can monitor and control your process from any workstation (node) on the network. You can use the Lookout networking capabilities to:

- View the same or different screens simultaneously on separate nodes,
- Make setpoint adjustments from any node,
- Acknowledge alarms from any node,
- Configure specific nodes for monitoring only,
- Configure a peer-to-peer architecture,
- Configure a client-server architecture,
- Configure network nodes to communicate via standard telephone lines, and
- Configure network nodes to communicate via radios.

Networking multiple Lookout nodes is a powerful and advanced capability. This chapter and the examples in it refer to many concepts and terms explained elsewhere in the manual. Before attempting to work with networking, make sure you have a strong understanding of the Lookout architecture and are comfortable working within Lookout on a single node.

To network using Lookout you must meet the following requirements:

- You must purchase and install on each computer a separate, licensed copy of Lookout.
- You must equip each computer on the network with compatible network hardware.
- You must install Microsoft Windows for Workgroups, Windows 95, or Windows NT on each computer.

Because Lookout is a native Windows application, it supports all the networks that Microsoft Windows supports, including Microsoft Windows Network, Microsoft LAN Server, Novell NetWare, Banyan VINES, 3Com 3+Open, DEC Pathworks, IBM LANServer, and so on.

Networking Considerations

Networking with Lookout is just passing data back and forth between two or more Lookout nodes. In Lookout, data is passed between and contained within objects. However, there are two basic methods for passing data between multiple Lookout nodes. The first method is *Multilink Networking*, which requires a NetDDE (Network Dynamic Data Exchange) link for each and every value being passed between the nodes.

The second method is *Table Networking*. Table Networking implements a *data concentrator* at each node where only tables (which concentrate the data) are linked between nodes. Although both methods involve connecting objects over a network using NetDDE, the implementation and total effect can be quite different.

If you plan to use one of the NetDDE methods, you will have to configure automatic NetDDE activation and possibly Trusted DDE Shares. See the *Networking with DDE* section in this chapter for setup instructions.

There is one more method for implementing network-like capabilities between multiple Lookout nodes. Unlike the first two methods, however, this technique does not rely on data being passed between Lookout nodes. Instead, you use PLCs, RTUs, or I/O to share data between Lookout nodes. This method is called *Hardware Networking*.

Multilink Networking

Multilink networking consists of passing data from an object on one computer to another object on a different computer. Once linked, any change to the value of one object is instantly propagated to and reflected by the second object on the other computer.

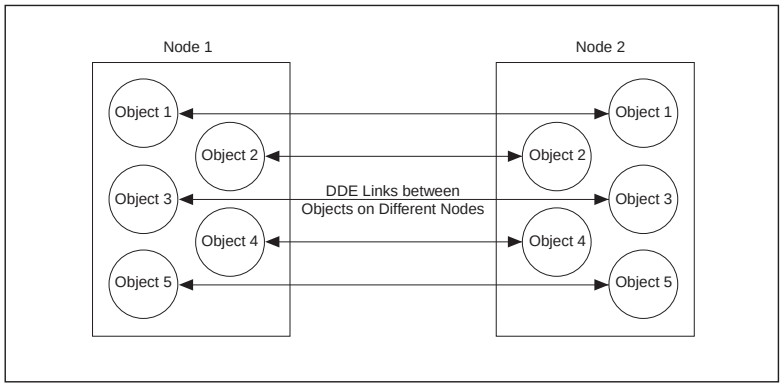


Figure 14-1. Multilink Networking

The way in which data is passed between two objects across a network depends on whether or not they are controllable objects.

Linking Controllable Objects

User-controlled objects have a DDE option. These objects include:

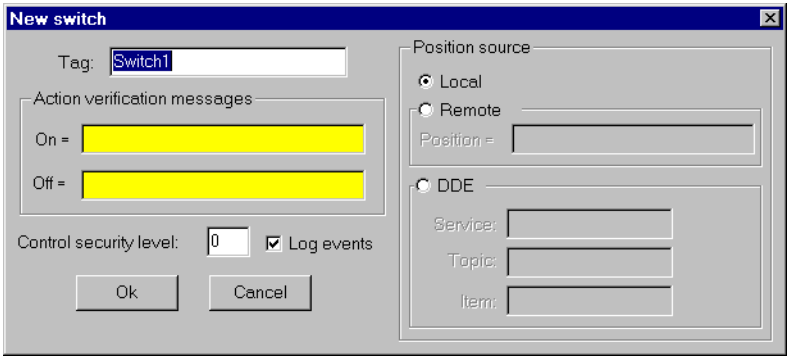
- Pot objects,
- Switch objects
- Pushbutton objects.

Controllable objects each have three **Position source** options: **Local**, **Remote**, and **DDE**. When you choose DDE, you instruct the object to take its value from a remote source. This could be a cell in a spreadsheet, another DDE aware application, or a second copy of Lookout running on the network.

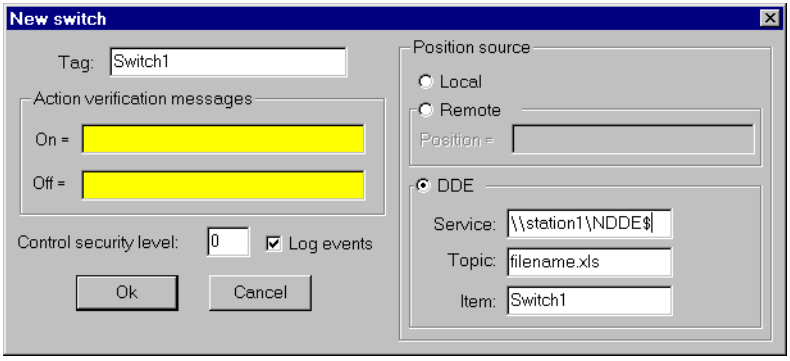
Linking Controllable Objects Together Across a Network

You need at least two computers connected by a functioning network to use the following example. This example refers to the two computers as Station1 and Station2.

1. At Station1, create a Switch object called `Switch1`. Leave **Position source** at the default setting of **Local**.



2. At Station2, create a corresponding Switch object called `Switch1`. However, change the Position source to **DDE**. (Although you are not required to give objects matching tagnames over a network, it makes your documentation easier to follow, and your applications easier to maintain.)



3. In **Service**, enter `\\Station1\NDDE$`, where `Station1` is the network name of the other computer.

   `Station1` is the network name of the other computer. The backslashes, `NDDE`, and dollar sign are required by Microsoft Windows.

4. In **Topic**, enter `Filename$`, where `Filename` is the name of the Lookout process file on the other computer.

   Again, the dollar sign is required but the `.LKP` file extension is not.

5. In **Item**, enter the name of the object at the other computer that you want to link to (in this case, the other object is also called `Switch1`).

6. Click on **OK**.

Test your network by flipping the switch. When you toggle the switch at either station, it instantly flips at the other workstation. You can add to, modify, delete, or revise your logic on either computer, completely on-line, and still maintain the network link between the two objects.

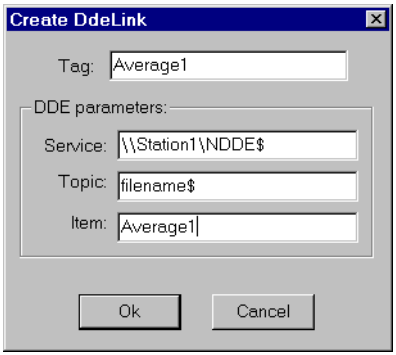
Linking Non-Controllable Objects

If a Lookout object is not a Pot, Switch, or Pushbutton, it probably does not have built-in DDE capabilities. However, all you are really trying to do is pass data between two computers. Once the data is in Lookout, any object can access it, so you can create a DdeLink object to receive data from another application using DDE or NetDDE. See the discussion of the *DdeLink* object in Chapter 18, *Object Class Reference*, for more information. In this example, the other application is Lookout, running on a second computer.

To Access Real-Time Data at Another Computer

You need at least two computers connected by a functioning network to follow this example. This example refers to the two computers as Station1 and Station2.

1. At Station1, create an Average object. It can average whatever values you want, but give it the tagname `Average1`.
2. At Station2, create a DdeLink object and name it `Average1`. Although you are not required to give objects matching tagnames over a network, it makes your applications easier to maintain.



3. In **Service**, enter `\\Station1\NDDE$`, where `Station1` is the network name of the other computer.

   `Station1` is the network name of the other computer. The backslashes, `NDDE`, and dollar sign are required by Microsoft Windows.

4. In **Topic**, enter `Filename$`, where `Filename` is the name of the Lookout process file on the other computer.

   Again, the dollar sign is required but the `.LKP` file extension is not.

5. Enter the name of the object at the other computer that you want to link to in the **Item** field (in this case, it is also called `Average1`).
6. Click on **OK**.
7. Finish the definition of the DdeLink object by inserting the DDE expression into your panel.

After you establish the link, you can connect other objects at Station2 to the `Average1` DdeLink object. Remember, just because you named the DdeLink object `Average1` does not make it a Lookout Average object. It just receives data from the Average object at Station1.

Table Networking

Table networking consists of consolidating all the data that needs to be shared over the network in a single table or array of values. You might think of these tables as "data concentrators"—although they can do much more than just serve as a repository for data. See the discussion of the *DataTable* object in Chapter 18, *Object Class Reference*, for more information. You can link a table via NetDDE to a corresponding table on another computer. Once linked, the tables update each other on any and all changes within their databases.

☞ **Note** ***Table networking is one of the most involved and advanced concepts in Lookout. You should have a complete understanding of objects, database connections, events, and general Lookout architecture before attempting table networking.***

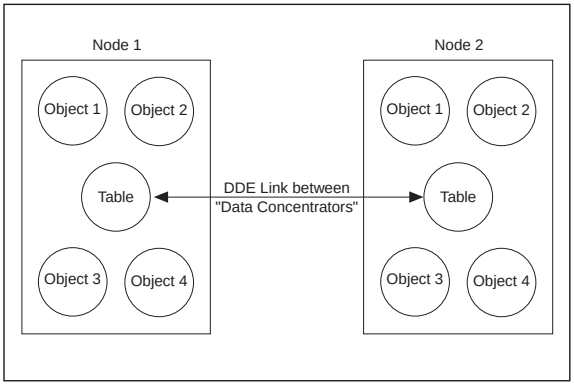
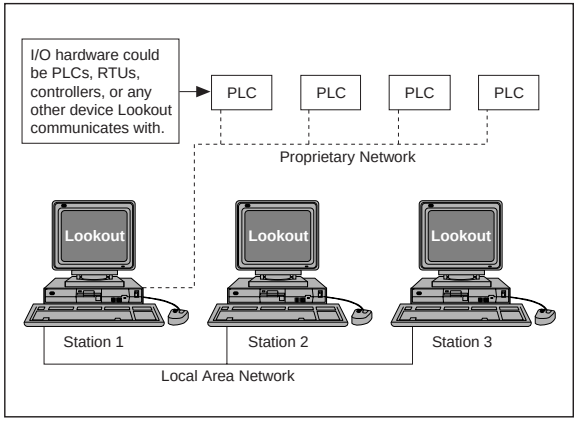


Figure 14-2. Table Networking

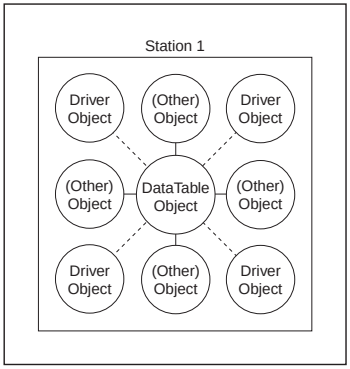
Linking multiple DataTables for networking purposes is simple enough if you plan before jumping headfirst into your application. The following example shows how table networking is done.

This example consists of three computers, each running a copy of Lookout. Set up three nodes on the network with the names Station1, Station2 and Station3. Station1 is the only node directly connected to a PLC, but you want to monitor and control the I/O from any of the three nodes. The following diagram depicts the network structure of this example.



By doing a little planning, you can configure Lookout at Station1 so that you can reuse its process file at Station2 and Station3. The following diagram depicts the general design of the Lookout process file at Station1. Notice that driver objects are connected to the DataTable object using

dotted lines while all other objects are connected using solid lines. This is for illustrative purposes only.



The focus of this picture is the driver objects and their connections to the DataTable. Eventually you will modify this process file to run at Station2 and Station3 by deleting all connections to Driver objects, and then deleting the driver objects themselves at Stations 2 and 3.

It is important to route all signals to and from driver objects through the DataTable. If you want to display an analog input from a PLC on a control panel, do not connect the PLC signal directly to the control panel. Instead, first connect it to the DataTable and then connect the appropriate DataTable cell to the control panel. If you need to connect an operator setpoint to an output on a PLC, do not connect the setpoint directly to the PLC. First connect the setpoint to the DataTable and then connect the appropriate DataTable cell to the output on your PLC. This is only necessary for signals that originate from, or are destined for, a driver object. Although this procedure adds another step to your Lookout configuration for Station1, it greatly simplifies the development of the process files for Station2 and Station3.

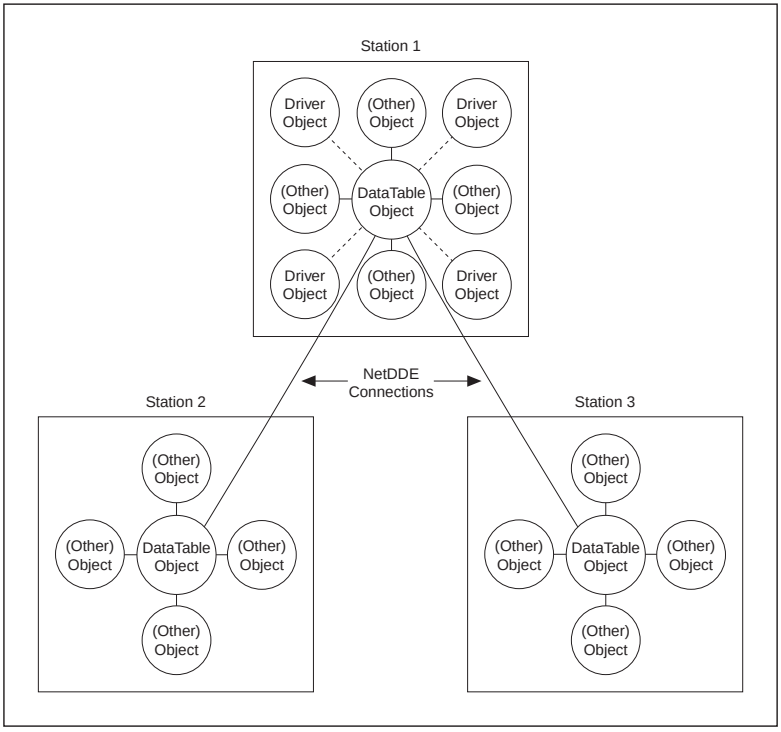
Now that you have developed an application at Station1, move over to Station2. Copy the process file to the Station2 computer and delete all the connections to and from your Driver objects. Then delete the Driver objects themselves.

Remember, you do not want Station2 attempting to communicate with the PLCs because there is no physical connection to that proprietary network. Station2 should get its information from Station1.

The only thing left to do is modify the DataTable object at Station2. Change its **Table location** parameter to **DDE**. In **Service**, enter `\\Station1\NDDE$`. In **Topic**, enter `Filename$`, where `Filename` is the name of the process file at Station1. In **Item**, enter the tagname of the DataTable object at Station1. Your DataTables are now bidirectionally connected through a NetDDE link.

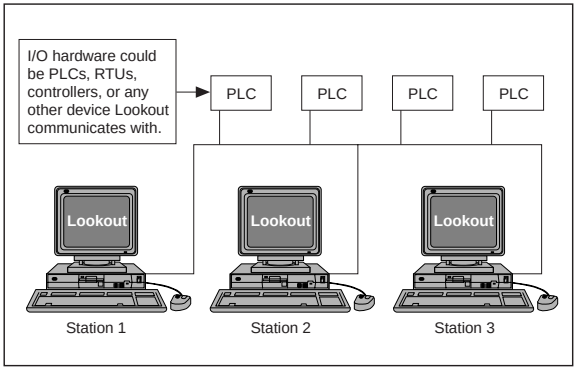
At this stage, you can easily add more operator stations to the network. Copy the process file from Station2 to Station3, or to any number of other Lookout nodes on the network. No changes to the files themselves are necessary.

Your Lookout network topography should now look something like this.



Hardware Networking

There is one fundamental concept that must be present in order to implement the Hardware networking method. *All* Lookout nodes you want networked must have direct communication access to *all* your hardware as displayed in the diagram below. Because of this, the Hardware networking method is only applicable in certain circumstances.



This is the most straightforward of the three networking methods to implement. Configure a process file at one node and copy it to all other nodes that have direct communication access to your hardware. The only thing to remember when building your process file is to configure all Pots, Switches, and Pushbuttons with the **Remote** parameter setting. This ensures that they stay in sync with the point they are writing to. In other words, if a switch at one Lookout nodes changes the status of an output, all the respective switches at the other Lookout nodes will reflect the change and automatically flip to stay in sync. See the *Pot*, *Switch*, and *Pushbutton* descriptions in Chapter 18, *Object Class Reference*, for more information on the **Remote** parameter.

Your hardware, network, and communications topology are the determining factors on whether you can use hardware networking. The hardware networking method is seldom available because it is rare that all your Lookout nodes can communicate directly with all your hardware.

Multilink and Table Networking Comparison

There are advantages and disadvantages to both Multilink and Table networking. Multilink is quick and easy for small jobs. However, every NetDDE link requires processor time, so the more links you have, the more CPU overhead is used. Remember, each value passed between computers requires a separate DdeLink object. Your computer can quickly become bogged down if hundreds of links are constantly sending values back and forth.

On the other hand, while the Table method may require more initial planning and development time, there will be only one NetDDE link between computers connecting the DataTables. This method dramatically reduces the amount of DDE traffic on the network. That is why table networking is the method of choice for large systems where computers are sharing hundreds or thousands of values across the network.

So how many points are too many? There is no single answer, but Lookout has been used with Table Networking method on systems having over 6000 values, and the network update time is well under one second with only three percent CPU usage.

Networking with DDE

Lookout uses Network DDE (Dynamic Data Exchange) for client/server networking.

Two components of Microsoft Windows are required when using Network DDE: the Network DDE agent (`NETDDE.EXE`) and the NetDDE Share Database Manager (DSDM). Exactly how these components are used depends on the operating system on which Lookout runs: Windows for Workgroups, Windows 95, or Windows NT.

The program `NETDDE.EXE` is an agent that runs in the background and transfers DDE messages between computers. This program should be running before an instance of Lookout is started if you intend for that session to act as a server. Lookout does not automatically run `NETDDE.EXE` itself.

Running NETDDE.EXE Automatically

The following three sections show how to make sure that NETDDE.EXE runs every time you start Windows.

Windows for Workgroups

1. In Microsoft Windows, choose the Control Panel icon or run CONTROL.EXE using the Program Manager **File»Run...** dialog box.
2. In the Control Panel window, choose the Network icon.
3. Click on the Startup option.
4. In the Startup Settings dialog box, make sure that the **Enable Network DDE** box is checked.
5. Double-click on **OK** and **OK** again to close the Control Panel.

Windows 95

1. In Windows 95, Click **Start»Settings»Taskbar...**
2. In the Taskbar Properties dialog box, choose the **Start Menu Programs** tab.
3. Click on the **Add...** button.
4. In the **Command line** field of the Create Shortcut dialog box, enter NETDDE.EXE; then click **Next**.
5. Choose the **Start Menu»Programs»StartUp** folder, then click **Next**.
6. Select a name (such as NetDDE) for the shortcut; select **Finish** and **OK**.

Windows NT

1. In the Program Manager, double-click on the Control Panel icon, or run CONTROL.EXE using the Program Manager **File»Run...** dialog box. If you are using NT version 4.00, click **Start**, point to **Settings**, and choose the Control Panel.
2. In the Control Panel window, double-click on the Services icon.
3. From the list of services, select Network DDE.
4. Click on the **Startup...** button.
5. In the **Startup Type** box, select Automatic.
6. Select **OK** and **Close** to exit the Control Panel.

Adding a Trusted DDE Share

To allow someone else to connect to a DDE share (for example, `process$`) on your computer when you are logged in, you must *trust* the DDE share. When another person connects to the share remotely, the application he connects to is running in your security context, not his own, because you are the logged-on user. You must give permission for him to access the share. Even another person who is an administrator cannot trust a share for your account.

In Windows for Workgroups and Windows 95, trusted shares are automatically created by Lookout.

In Windows NT, you must use the program `DDESHARE.EXE` to add a trusted share. Exactly how you set this up depends on the level of security you require. The following sequence of steps describes how to set up a share that gives everyone on the network full DDE access to Lookout:

1. From the Program Manager, select **File»Run...** If you are using NT version 4.00, click **Start** and select **Run**.
2. Enter `DDESHARE.EXE`.
3. Select **Shares»DDE Shares...**
4. In the DDE Shares dialog box, select **Add a Share...**
5. In the DDE Share Properties dialog box:
    a. For the **Share Name**, enter the name of the Lookout process file that you want to share, omitting the `.LKP` extension, and adding a '`$`' at the end. For example, if your process file was `PROCESS.LKP`, you would enter:

       `PROCESS$`

       on this line.
    b. For the **Static Application Name**, enter `Lookout`.
    c. For the **Static Topic Name**, enter the process file without extension, such as `PROCESS`.
    d. Leave the **Allow start application** and **is service** boxes unchecked.
    e. In the **Item Security** box, select the `Grant access to all items` box.
    f. Click on the **Permissions...** button.

g. In the DDE Share Name Permissions dialog box:
   - In the **Names** list, select `EVERYONE`. If there is no entry for `EVERYONE`, use the **Add...** button and associated dialog box to create it.
   - In the **Type of Access** box, select `Full Control`.
   - Select **OK**.

h. Select **OK**.

6. You should now be back in the DDE Shares dialog box. Select the share you just created and click on the **Trust Share** button.
7. In the Trusted Share Properties dialog box, select the **Initiate to Application Enable** button; click on **OK**.
8. In the DDE Shares dialog box, click on **OK**.
9. Exit the DDE Shares program.

☞ **Note** ***In order for the DDE share to be available, it must be trusted by the user currently logged in to the computer (using steps 6 through 9). Alternatively, the share may be provided as a service by selecting the "Is service" box in the "DDE Share Properties" dialog box. In this case, the share is always available to anyone listed in the "DDE Share Name Permissions" dialog box.***

☞ **Note** ***When a Lookout client tries to connect to a DDE server on a computer running NT, it may query the user for a user name and password that are valid on the server computer. In this case, the user password should not be NULL, or the attempt to connect to the server will fail.***

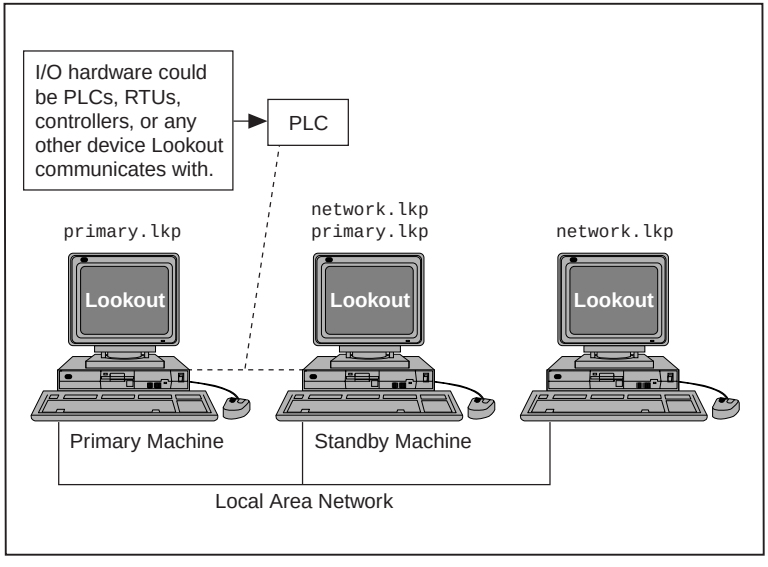
15

Redundancy

With Lookout you can configure two process control computers for redundancy, providing automatic transfer of control if the primary computer should fail.

☞ **Note** ***This chapter pertains to systems that implement networking of multiple Lookout nodes. It is not necessary to read this chapter for stand-alone systems. You should understand process files (*** .LKP***) and state files (*** .LST***) before using attempting to use this chapter.***

You can set up computer redundancy using the Lookout *standby* feature. Standby refers to a network configuration in which one computer is designated the *primary* computer and another is designated the *standby* computer. The primary computer normally monitors and controls the process while the standby computer monitors the primary computer. If the standby node fails to receive a *heartbeat* from the primary node within a certain time period, the standby computer automatically takes over and assumes the role of the primary computer. The following figure depicts a typical Lookout network structure with the standby option configured.



Standby Basics

There are three basic principles assumed in the diagram above.

First, both the primary *and* standby computers must have direct access to your field I/O. If the primary computer fails for any reason, you cannot rely on it anywhere in your backup strategy. Your standby computer must have the same direct access to all your I/O, because it will assume the responsibility of the primary computer. There are several ways to accomplish this, depending on your hardware, network, and communications topology. Call National Instruments technical support for suggestions if you need help.

The second principle is that you can designate any node on the network as the standby computer. It does not need to be the closest physical computer on the network, though this is typically the case due to the restraints imposed by the first principle. During normal operations (that is, while the primary computer if functioning properly), the standby computer is identical to all other nodes on the network. In other words, Lookout is running identical process files on all network nodes (`NETWORK.LKP` in this example). However, the standby computer should also have a copy of the primary computer process file in its Lookout directory, `PRIMARY.LKP` in this example. As soon as the standby computer fails to receive a scheduled watchdog signal from the primary, it automatically closes the `NETWORK.LKP` process file and opens the `PRIMARY.LKP` file. From that point on, the standby node assumes the role of the primary computer and all other Lookout nodes automatically recognize this fact.

☞ **Note** ***When a failover occurs, (that is, when the standby computer takes control of the process), all Lookout Network DDE links from other computers to the primary computer are automatically redirected to the standby computer.***

The third and last standby principle takes no additional configuration on your part, but is important to understand for peace of mind. A stand-alone Lookout application is responsible for periodically updating its own state file. In a standby system configuration, the primary computer not only updates its own state file, but also the state file of the standby computer. When a failover occurs, the standby computer takes over where the primary computer left off, ensuring bumpless transfer of control.

Failover Scenarios

There are four reasons for the standby computer to take control of the process:

- The primary computer is down (that is, it is not turned on, Windows is not running, or an application fault has locked the computer).
- The primary computer is running, but it is not running Lookout.
- The primary computer is running Lookout, but it is not running the correct process file.
- The primary computer is running the correct process file within Lookout, but the two computers cannot communicate over the network.

The purpose of the standby feature is to provide automatic failover to a second computer. It does not automatically return to its previous primary-standby status once your network or other problems are corrected. Actually, the standby computer will automatically switch back from the primary process file to the standby process file as soon as it detects that the primary computer is back up and running. You should, however, copy the state file from the standby computer back to the primary computer before restarting Lookout on it if you want a bumpless transfer.

To restore your Lookout network to its original setup (that is, with the primary computer running its `PRIMARY.LKP` file and the standby computer running its `NETWORK.LKP` file), first ensure bumpless transfer by copying the state file from the standby computer back to the primary computer. Do this before you restart Lookout on the primary computer if you want to have the settings in their current state.

Configuring Standby

Follow this general procedure to implement the Standby feature:

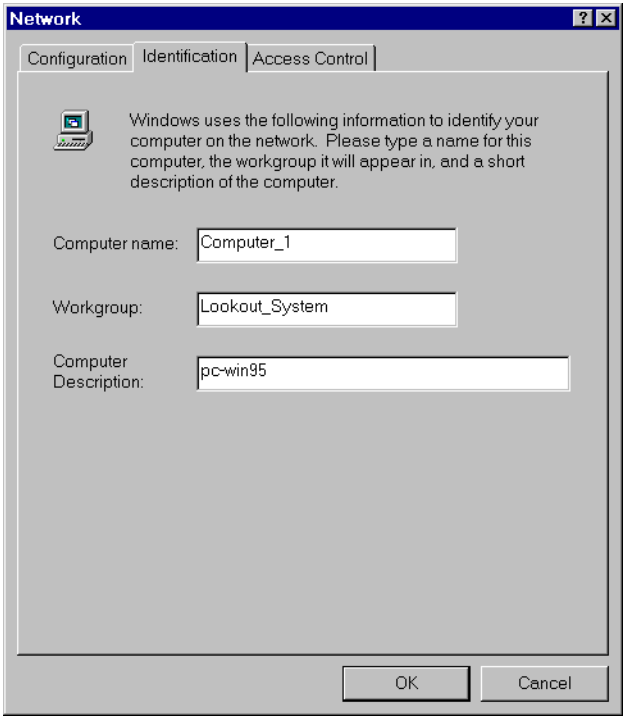
1. Define network settings for both computers. See the *To Define Network Settings* section below for information on how.
2. Ensure that the primary computer has network access to the Lookout directory on the standby computer hard drive. See the *To Enable File Sharing in Windows 3.1.1* and *To Enable File Sharing in Windows 95 and Windows NT* sections of this chapter for information on how.
3. You must enable file sharing in order for Lookout on the primary computer to update the state file on the standby computer.

4. Copy the process file from the primary computer to the Lookout directory on the standby computer. In the previous example this file was named `PRIMARY.LKP`.
5. Configure Lookout standby settings for both computers. See the *To Configure Standby Options* section of this chapter for information on how.

To Define Network Settings

On both the primary and standby computers:

1. In Microsoft Windows, open the Control Panel.
2. In the Control Panel window, choose **Network**.
3. In the **Network** dialog box, assign the computer a name.



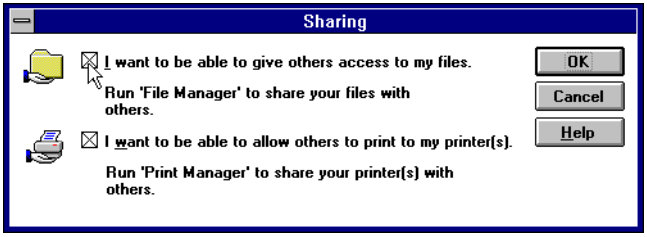
4. Make the **Computer Name** of the primary and standby computers consecutive. Give the primary computer the first name of the pair, and the standby computer the second name of the pair. For example, `XXXA` and `XXXB` or `System01` and `System02`. Notice in the examples that the names are identical except for the last character.

5. Make sure both computers belong to the same **Workgroup**.
6. In the Windows 3.1.1, choose the **StartUp option**.
7. In the **StartUp Settings** dialog box, check the **Enable Network DDE** check box.
8. Choose **OK** to accept your settings.

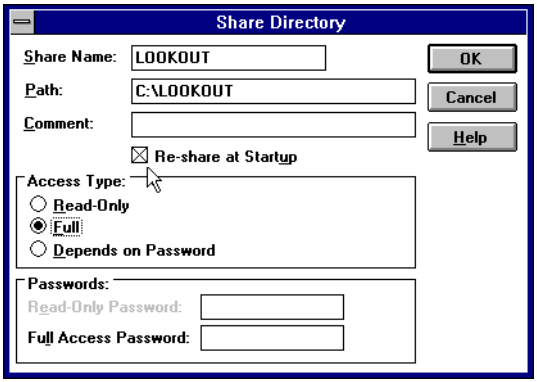
If you change the settings, the computer prompts you to restart the computer.

To Enable File Sharing in Windows 3.1.1

1. Configure the *standby* computer to be able to share files:
   a. In Microsoft Windows 3.1.1, choose the Network Setup icon.
   b. In the **Network Setup** dialog box, choose Sharing.
   c. In the **Sharing** dialog box, check the box enabling file access by others.


Sharing
I want to be able to give others access to my files.
Run 'File Manager' to share your files with others.
I want to be able to allow others to print to my printer(s).
Run 'Print Manager' to share your printer(s) with others.
OK
Cancel
Help

   If you change the settings, the computer prompts you to restart Windows.
2. Configure the *standby* computer so that the Lookout directory on its hard disk can be shared with the primary computer:
   a. In Microsoft Windows, select the file manager icon.
   b. In the File Manager window menu bar, select **Disk»Share As...**.



c. In **Path**, enter the path name of the Lookout subdirectory. For example, enter `C:\LOOKOUT`.

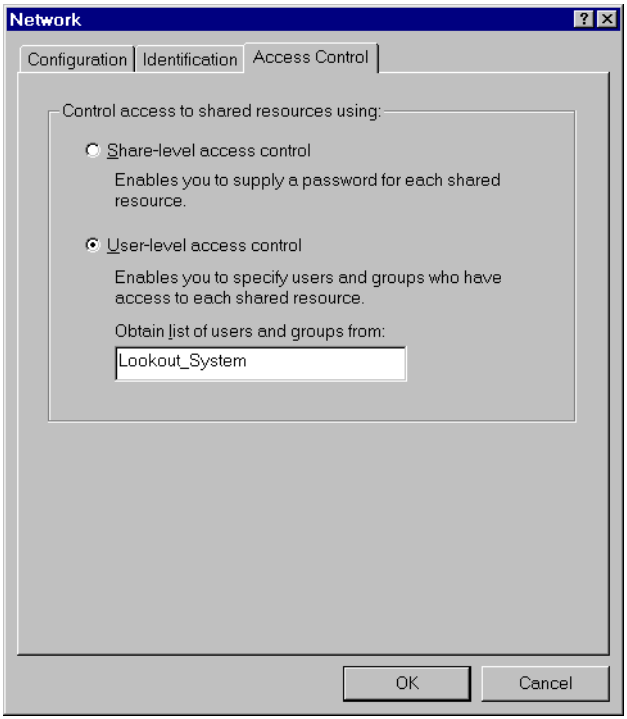
d. In **Share Name**, enter the name you want to assign to this directory. Make the name recognizable. In this example, call the directory `LOOKOUT`.

e. In the Share Directory dialog box, ensure that the check box enabling **Reshare at Start-Up** is selected.

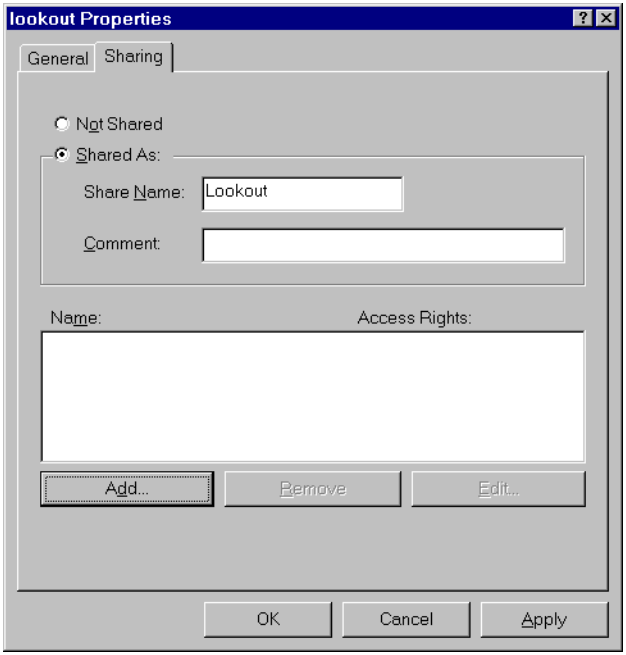
f. In the Share Directory dialog box, ensure that the check box enabling **Full** access is selected and select **OK**.

To Enable File Sharing in Windows 95 and Windows NT

1. Configure the *standby* computer to be able to share files using the sharing options in the Network control panel.



2. Select the Lookout directory on the standby computer and configure it to be shared.

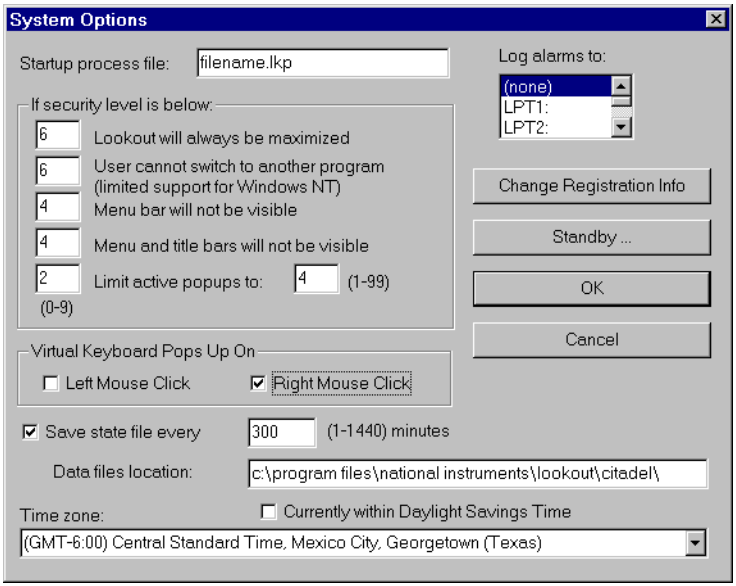


3. Using the Add option, add the authorized computers or users you want to be able to share to. Make sure full access privileges are granted.

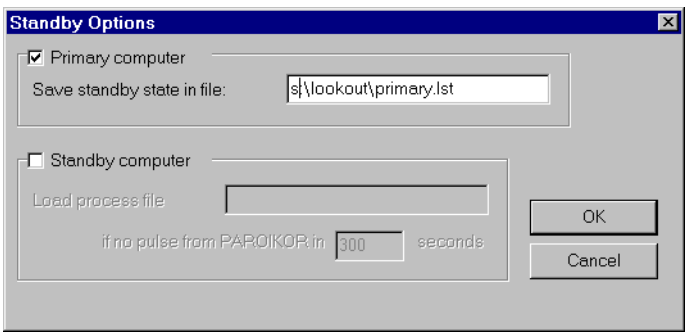
To Configure Standby Options

1. On the *primary* computer:
   a. From the Lookout menu bar, select **Options»System...**.
   b. In the **System Options** dialog box, select the check box to enable Lookout to **Save state data every ___ minutes**.

      You can change the frequency if desired.

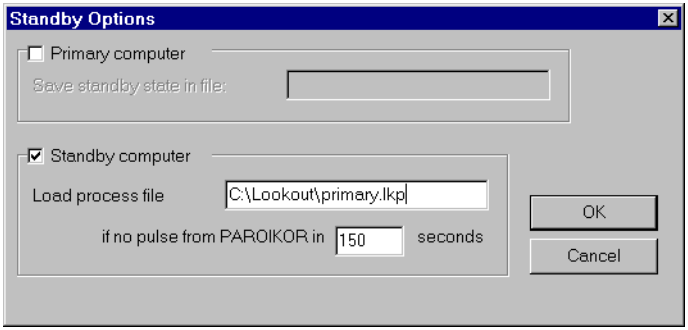


2. In the **System Options** dialog box, select the **Standby** button. You should now see the **Standby Options** dialog box.
   a. In the **Standby Options** dialog box, choose **Primary computer**.
   b. In the **Save standby state in file** data field, enter the full path name to where you want to save a duplicate copy of the state file on the standby computer. In this example, the Workgroup network settings are defined such that drive `S:` on the primary computer represents drive `C:` on the standby computer.



   c. Select **OK** to save your changes.

3. On the *standby* computer:
   a. In Lookout, select **Options»System...**
   b. In the **System Options** dialog box, select the **Standby...** button.
   c. In the **Standby Options** dialog box, choose **Standby computer**.



   d. In the **Load process file** data field, enter the full path name of the process file you want Lookout to automatically load if a failover situation should occur.

      To continue the example, enter:

      `C:\LOOKOUT\PRIMARY.LKP`

      Notice that the path name is different from what you entered in Step 1e. This is because you are now at the standby computer, referencing the local hard disk.

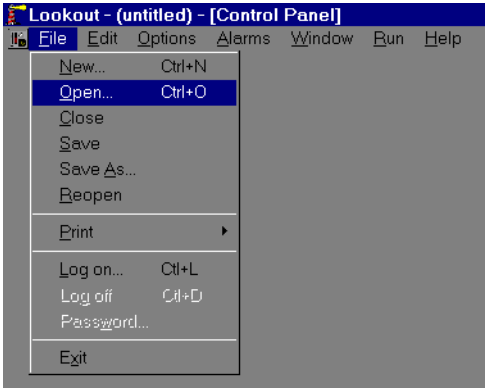
   e. In the **if no pulse from...** data field, enter the desired frequency of the watchdog timer. The standby computer loads the identified process file and assumes control of your process if no pulse is received from the primary computer within the watchdog timeout period.

   f. Select **OK** to save your changes.

16

Runtime Menu Commands

This chapter describes Lookout menu bar pull-down commands available in Normal mode (that is, not Edit) mode.



Every menu command has a predetermined security level associated with it. Only operators whose security levels are equal to or greater than that of a particular command can access that menu command. See Chapter 10, *Security*, for more information on security.

File Commands

File»New

Security Level: 9
Shortcut Keys: <CTRL+N>

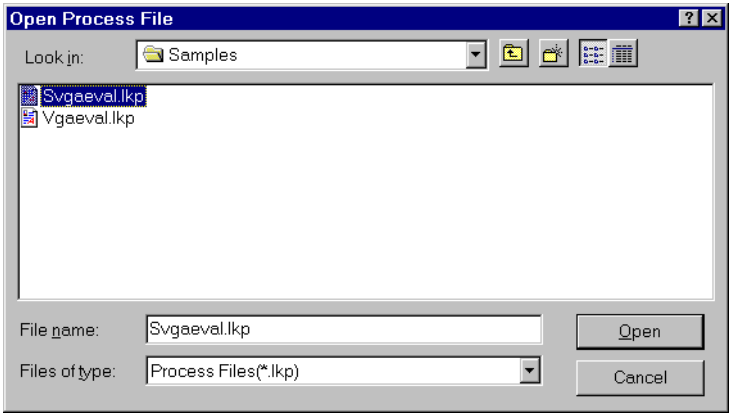
The **File»New…** command is the first step in creating a new process file. It is important to notice that you must use the **File»Save…** or **File»Save As…** command to save your work and create a new process file.

File»Open

Security Level: 8
Shortcut Keys: <CTRL+O>

Use the **File»Open…** command to open a process file for execution. Lookout can only execute one process file at a time. If there is already a process running, Lookout asks if you want to abort the currently executing process before it opens a new process file.

When you select **File»Open…**, a dialog box pops up that you can use to select the current disk drive/directory and scroll through a list of process files. Once you find the process file you want to open, click on the file name in the file list box or type the file name and press the **OK** button, or just double-click on the file name in the list box.

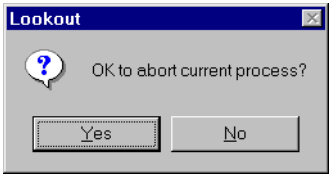


If a process file is already open when you select a new file, a message box pops up asking if you want to abort the current process. The current process continues to execute until you answer yes.

File»Close

Security Level: 8
Shortcut Keys: none

The **File»Close** command halts execution of the currently loaded process file. If you made changes to your process file and have not saved your work, Lookout first prompts you to do so before letting you close your process.



The process continues to execute until you answer Yes.

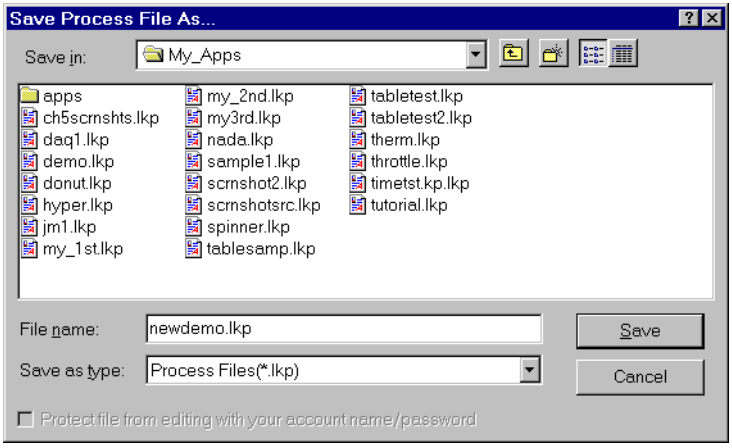
Caution ***The File»Close command shuts down process execution—be sure that this is what you want to do before selecting this command. Your Lookout application may be controlling critical processes and shutting it down could cause serious problems.***

File»Save

Security Level: 9
Shortcut Keys: none

The **File»Save** command saves the currently executing process to disk. Lookout replaces the old process file on disk with a new version that reflects all changes made: new or deleted objects, different display characteristics, new or modified I/O connections, setpoint adjustments, and so on. This includes saving changes to the `.LKP`, `.LKS`, and the `.LST` files. See Chapter 3, *Getting Started*, for more information about these files.

If you are creating a new process file from scratch, the **File»Save** command invokes the **Save Process File As…** dialog box, prompting you for instructions on naming the file.



File»Reopen

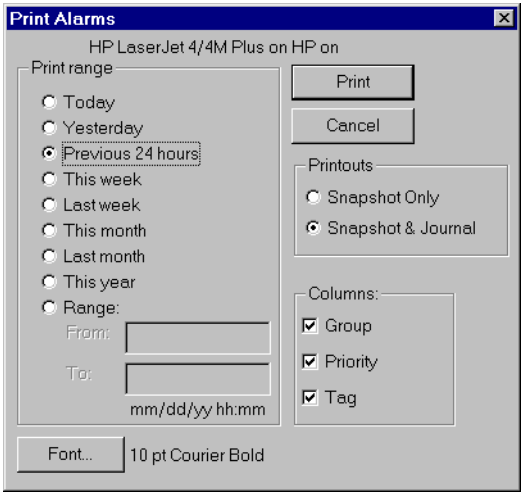
Security Level: 8
Shortcut Keys: none

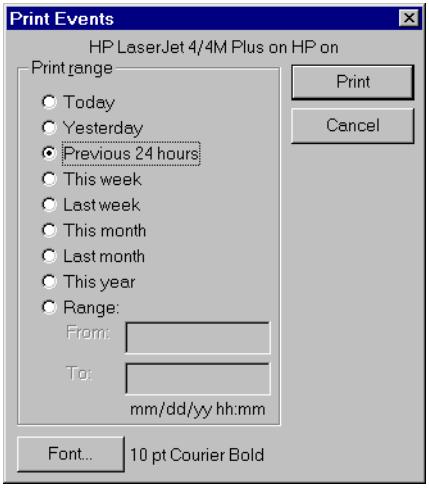
The **File»Reopen** command reloads the currently executing process from disk. This in turn clears all alarms from the alarm window. All trends and setpoints, however, retain their previous values.

File»Print»Alarms
File»Print»Events

Security Level: 1
Shortcut Keys: none

Use the **File»Print** commands to print alarm and event reports. When you select one of these commands, a dialog box appears, prompting you to specify the time span to be printed. Use the **Print Range** option to specify any arbitrary time span (in military time). Notice that the Alarm and Event dialog boxes are slightly different. See Chapter 9, *Alarms*, for more information on printing alarms.

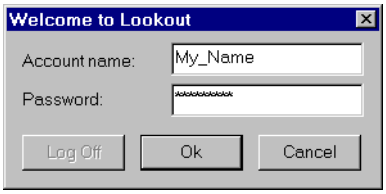




File»Log on

Security Level: 1
Shortcut Keys: <CTRL+L>

Use the **File»Log on…** command to log on to Lookout as the current operator/integrator with your predefined account name and password—although Lookout does not necessarily require a password. See Chapter 10, *Security*, for more information.



Only one person can log on at a time. You can also initiate this command by clicking on the account box in the status bar near the bottom left hand corner of the screen, where the account name appears.

File»Log off

Security Level: 1
Shortcut Keys: <CTRL+L>

The **File»Log off** command instantly logs off the current account name and leaves (nobody) logged onto the system. You can also initiate this command by clicking on the account box near the bottom left hand corner of the screen, where the account name appears. When the **Welcome to Lookout** dialog box appears, select the **Log Off** button. See Chapter 10, *Security*, for more information.

File»Password

Security Level: 1
Shortcut Keys: none

Use the **File»Password…** command to change a password. People can easily change their own passwords as needed with this command. Lookout also keeps track of how old any given password is so that you can monitor password age. See Chapter 10, *Security*, for more information.

File»Exit

Security Level: 8
Shortcut Keys: none

The **File»Exit** command halts the executing process and exits Lookout. After selecting **File»Exit**, Lookout gives you the opportunity to save the process file and asks you to verify that you want to abort the current process.

Caution ***The File»Exit command shuts down process execution—be very sure that this is what you want to do before selecting this command. Your Lookout application may be controlling critical processes and shutting it down could cause serious problems.***

Edit Command

Edit»Edit Mode

Security Level: 9
Shortcut Keys: <CTRL+SPACE>

The Edit menu commands are only available when Lookout is in Edit Mode. You can use Edit Mode to create and/or modify a process file. It is important to notice that Lookout remains on-line, even when in Edit Mode! Take care when assigning security levels 9 and 10 to system accounts as anyone with these security levels has access to this command.

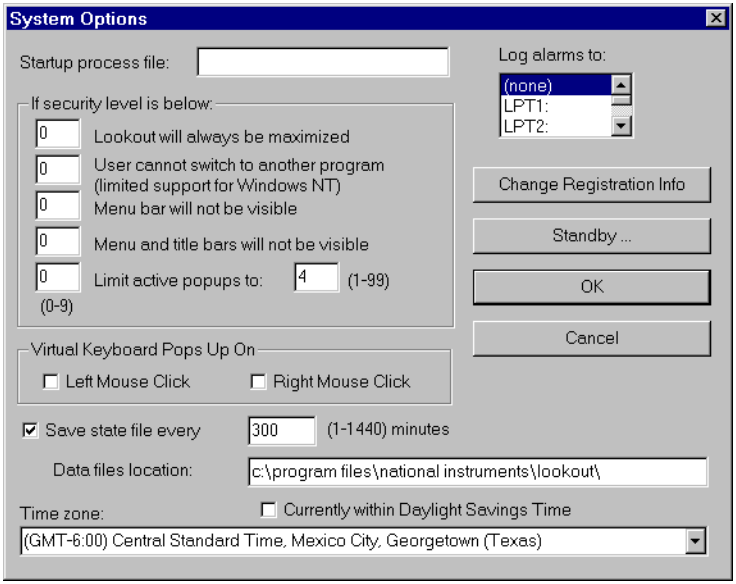
The menu bar at the top of the Lookout screen adds additional commands when you toggle into Edit Mode. You can use these additional commands for process configuration. See Chapter 17, *Edit Mode Menu Commands*, for more information.

Option Commands

Options»System

Security Level: 9
Shortcut Keys: none

The **Options»System…** command is used to configure various system-level operating parameters. The first parameter, **Startup process file**, instructs Lookout to automatically load a predetermined process file every time it starts. This command, used in conjunction with the method described in the Automatic Process loading section of Chapter 3, *Getting Started*, ensures that any time your computer reboots or Windows restarts, the appropriate application loads. You can also specify a particular Control Panel to maximize upon startup. See Panel object definition for additional information.



The second parameter is the **Log alarms to** setting that defines whether to send alarm records to the printer automatically as they occur—and if so, through which port. Alarm records include alarm activation, acknowledgment, and deactivation. Normally, this should be set to none and the Lookout **File»Print»Alarms…** command should be used to print custom alarm reports on demand.

Use the third set of parameters, in the **If security level is below** area, to define windowing limitations based on the security level of the user currently logged on. See Chapter 10, *Security*, for more information.

Set the **Virtual Keyboard Pops Up On** parameter to suit your personal preference.

The **Save state file every *N* (1-1440) minutes** parameter defines the frequency at which the Lookout State file (`.LST`) is updated. The Lookout state file contains setpoints and other important data values held within objects. It is normally updated when the Lookout application file is saved, closed or exited. However, hardware on which Lookout is installed may be vulnerable to power upsets. When there is a power loss, Lookout, like all software, immediately shuts down. In such cases, it does not have a chance to update the state file. This parameter tells Lookout to save data to the state file periodically. On subsequent restart, Lookout reinstates all of the setpoints and other important data values that were last captured in the state file.

Use the **Change Registration Info…** button to enter registration data, such as your key code. See Chapter 2, *Introduction*, for more information on *Registration*.

The **Standby…** button, configures your system for computer-level redundancy. See Chapter 15, *Redundancy*, for more information.

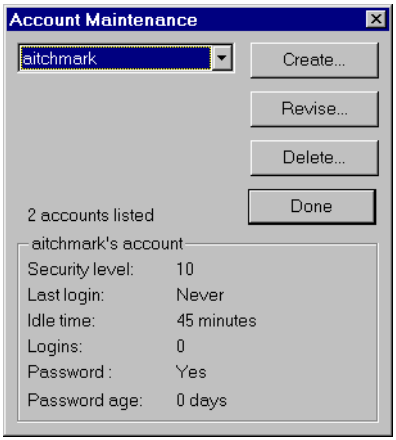
You can set the root directory where Lookout logs your spreadsheet, Citadel database, event files, and alarm files in this dialog box. In the figure, **Data files location** specifies `C:\program files\national instruments\lookout` as the root directory under which all data logging should begin. Depending on which logging techniques are implemented, Lookout may store the files in additional subdirectories under the specified root directory.

Use the **Time zone** setting to make sure Lookout saves data to the Citadel database with International Coordinated time stamps (Greenwich Mean Time.). Based upon your selection, Lookout automatically converts the International Coordinated time stamps back into local time when plotting trends. This ensures your database reflects the correct local time wherever you are using it. Be sure to select **Daylight Savings Time** when your region observes these clock adjustments.

Options»Accounts

Security Level: 10
Shortcut Keys: none

Use the **Options»Accounts…** command to create, revise, and delete system user accounts. See Chapter 10, *Security*, for more information on accounts.



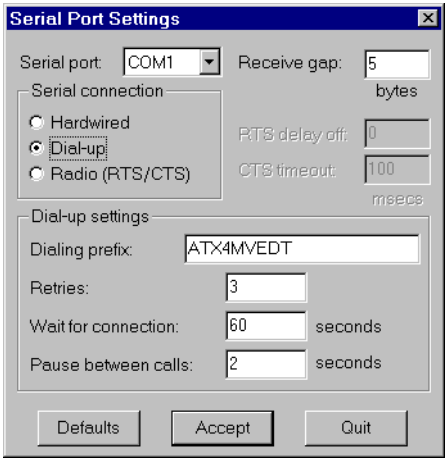
Options»Modbus

The **Options»Modbus…** command only appears when a Modbus object exists in the currently executing process file. This command calls an information box that describes the Modbus version settings and provides statistical data on protocol communications. See Chapter 18, *Object Class Reference*, for more information on the *Modbus* object class.

Options»Serial Ports

Security Level: 9
Shortcut Keys: none

Use the **Options»Serial Ports…** command to configure your computer serial port communication topology. Each serial port can have a unique setting: Hardwired, Dial up, or Radio. See Chapter 6, *Serial Communications*, for more information.



Options»Import APT Database

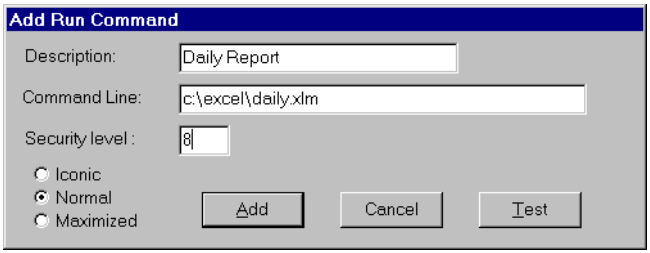
The **Options»Import APT Database…** command only appears if a Tiway object exists in the currently executing process file. Choose this command to import an APT tag file and create new database members for the selected Tiway object. See Chapter 5, *Developer Tour*, for information on object databases. Also see Chapter 18, *Object Class Reference*, for information on the *Tiway* Object Class and the *APT* database.

Run Commands

Run»Add

Security Level: 9
Shortcut Keys: none

The **Run»Add…** command adds up to ten DOS-style commands to the run submenu list. Operators subsequently invoke these commands with a click of the mouse—without leaving Lookout. For instance, you might want to print a custom report on demand. The following example loads Excel and runs the macro `daily.xlm`, which pulls historical data off the hard drive, places it into a preconfigured report template, and sends it to the printer.

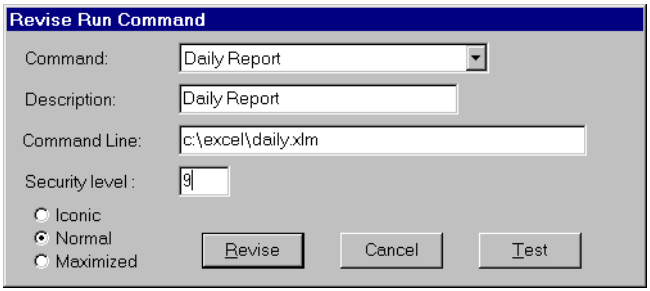


The **Security Level** setting applies to the any command you are add or revise. This means you can configure multiple commands, some of which may only be invoked by high level operators.

Run»Revise

Security Level: 9
Shortcut Keys: none

Use the **Run»Revise…** command to modify previously configured run commands.



Run»Delete

Security Level: 9
Shortcut Keys: none

Use the **Run»Delete…** to remove previously configured run commands.

Alarm Commands

Alarms»Show

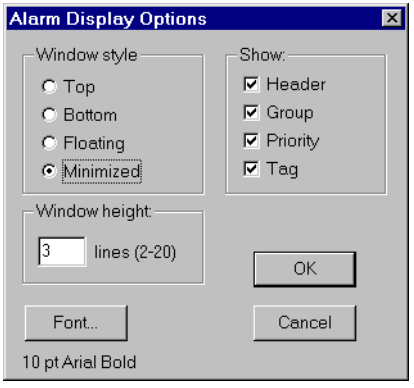
Security Level: 1
Shortcut Keys: <CTRL+A>

The **Alarms»Show** command instantly displays the alarm window as a floating style window if it is not already being displayed. You can select this command when you want to quickly and easily locate the alarm window from any location on any control panel. Notice the shortcut keystroke, <CTRL+A>. You can also call the alarm window by clicking on the alarm box in the far right end of the status bar at the bottom of the screen. See Chapter 9, *Alarms*, for more information.

Alarms»Display Options

Security Level: 1
Shortcut Keys: none

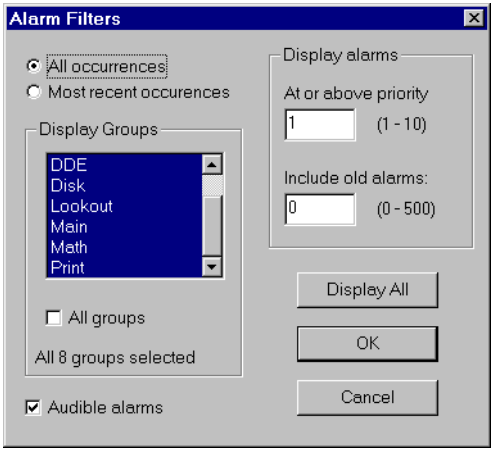
The **Alarms»Display Options…** command changes the display style of the alarm window (Top, Bottom, Floating). You can also use this command to modify internal alarm display formats such as font, header, and various alarm information. See Chapter 9, *Alarms*, for more information.



Alarms»Filter Options

Security Level: 2
Shortcut Keys: none

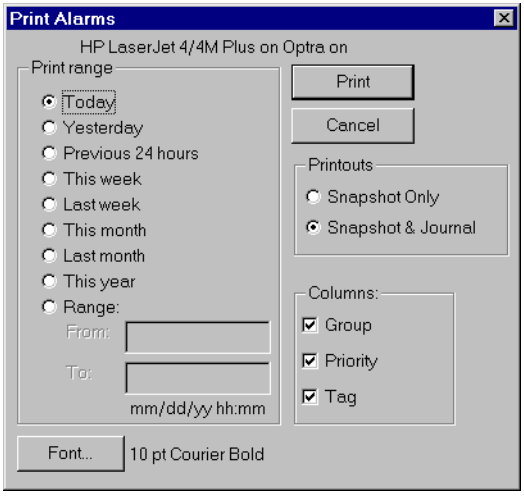
Use the **Alarms»Filter Options…** command to filter the alarms displayed in the alarm window by group, priority, and occurrence. See Chapter 9, *Alarms*, for more information on filtering alarms.



Alarms»Print

Security Level: 1
Shortcut Keys: none

The **Alarms»Print…** command invokes a dialog box you can use to specify the time period of the alarms you want to print See Chapter 9, *Alarms*, for more information on printing alarms.



Alarms»Select All

Security Level: 2
Shortcut Keys: none

The **Alarms»Select All** command is a shortcut method for selecting all visible or non-filtered alarms for acknowledgment. Alarms that are not visible because of filtering or because the alarm window is minimized will not be selected. This is especially useful if your process is experiencing high numbers of alarms. Selecting each alarm individually can be very time consuming. See Chapter 9, *Alarms*, for more information on acknowledging alarms.

Alarms»Deselect All

Security Level: 2
Shortcut Keys: none

This command is a shortcut method for deselecting all alarms that are currently selected for acknowledgment. If you want to deselect only specific alarms, click on the individual alarm line. See Chapter 9, *Alarms*, for more information on acknowledging alarms.

Alarms»Acknowledge

Security Level: 2
Shortcut Keys: none

Operators select the **Alarms»Acknowledge…** command to acknowledge alarms that are currently selected for acknowledgment. See Chapter 9, *Alarms*, for more information on acknowledging alarms.

Window Commands

Window»Arrange Icons

Window»Arrange Icons arranges the control panel icons across the bottom of the Lookout workspace in neat columns and rows. Lookout spaces them according to the setting for icon spacing in the Microsoft Windows Control Panel.

Window»Minimize All

Window»Minimize All minimizes all control panels and displays their icons across the bottom of the Lookout workspace.

Window»*nTitle*

The titles of all control panels appear at the bottom of the Window menu, each with a number you can use when selecting panels with the keyboard instead of the mouse. You can locate a particular panel by finding its title in this list and selecting it. Lookout then displays the panel in the workspace. Of course, if the panel is outside of the visible workspace, you may have to scroll around in the workspace to make the panel visible.

Window»More Windows

If you have too many control panels to fit in the window menu, the **More Windows** command appears at the bottom of the Window menu. The **Window»More Windows** command pops up a dialog box you can use to scroll through the control panel titles and select the window to display by double-clicking on the panel title.

17

Edit Mode Menu Commands

This chapter describes the Lookout menu bar pull-down commands available in Edit mode. You use Edit mode to perform all system configuration and editing. To toggle into and out of Edit mode, select **Edit»Edit mode**, or press <CTRL+SPACE>.

When you toggle into Edit mode, Lookout adds four additional menu pull-downs to the menu bar: **Insert**, **Object**, **Arrange**, and **Change**. In addition, it adds several submenu commands to the **Edit** pull-down menu. This chapter describes all the commands accessible while in Edit mode.

Note ***Lookout continues execution of the process file while in Edit mode. From the very beginning of developing a new process file, Lookout is on-line, running any control logic you have configured to that point. Create a new object, connection, or expression and Lookout immediately adds it to the currently running process file. Because Lookout remains on-line all the time, you never have to shut down the process to add or delete controllers, I/O points, or modify your logic.***

Caution ***Take extreme care when assigning security levels 9 and 10 to user accounts. Both of these security levels can access Edit mode and modify a process file!***

Edit Commands

Edit»Undo

Security Level: 9
Shortcut Keys: <CTRL+Z>

You can use the **Edit»Undo** command to undo the last ten (10) changes made to a control panel. You cannot, however, undo all changes. You can only undo graphical changes, not underlying object or logic modifications, such as connections or ranges.

For example, create several objects and insert them on a control panel. Add some text. Now move them around the screen, delete them, change fonts, colors, and so on with the **Change** menu command. Lookout keeps track of these graphical changes in a ten-deep stack. You can then select **Edit»Undo** and Lookout will undo the most recent change. Select **Edit»Undo** again

and Lookout will undo the next most recent modification, and so on. There are several things, such as invoking revise dialog boxes, that can wipe the Lookout undo buffer clean. Do not be surprised if you cannot undo something that happened several steps previous to your action.

Edit»Cut

Security Level: 9
Shortcut Keys: <CTRL+X>

The **Edit»Cut** command removes selected items from the control panel and moves them to the Windows Clipboard. You can paste these items back onto the same control panel or a different control panel.

Edit»Copy

Security Level: 9
Shortcut Keys: <CTRL+C>

The **Edit»Copy** command copies selected items from the control panel to the Windows Clipboard. You can then paste these items onto the same control panel or a different control panel.

Edit»Paste

Security Level: 9
Shortcut Keys: <CTRL+V>

The **Edit»Paste** command retrieves previously cut or copied information from the Windows Clipboard, and pastes it on the active control panel. You may repeatedly paste the same information from the Clipboard until you cut or copy new items into the Clipboard.

☞ **Note** ***When you cut or copy an item on a control panel, Lookout remembers the position (X-Y pixel coordinate) of the item relative to the source panel upper left corner. When you subsequently paste the item onto another control panel, Lookout pastes it at the remembered position. For this reason, when you paste an item onto a smaller panel, it might not be visible; rather its position may be outside the panel dimensions (to the right or below). You may have to resize the panel to see the pasted item.***

Edit»Delete

Security Level: 9
Shortcut Keys: <Delete>

The **Edit»Delete** command removes the current selection from a control panel. This command is disabled if nothing is selected. You can delete individual objects, graphics, and expressions or you can lasso large groups of items for deletion. If you delete something by mistake, select the **Edit»Undo** command and the deleted item reappears on the control panel.

Deleting an object from a control panel only erases the graphical representation of that object, not the object itself (see the **Object»Delete** command later in this chapter). For example, you can display on a control panel a pulse timer whose output signal is connected to another object. If you delete the timer display off the screen, the underlying timer object is still present, working in the background.

Edit»Select All

Security Level: 9
Shortcut Keys: none

The **Edit»Select All** command selects all the displayed entities on the active control panel.

Edit»Edit Mode

Security Level: 9
Shortcut Keys: <CTRL+SPACE>

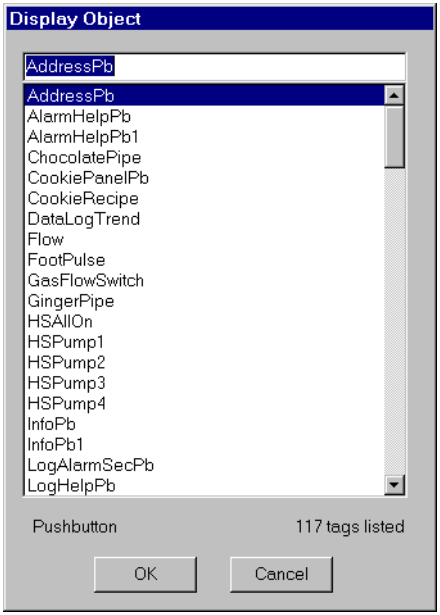
This command acts as a toggle switch you can use to flip into and out of Edit mode. Lookout only toggles to and from edit mode when a process file is running. When Lookout is in Edit mode, the status bar at the bottom of the Lookout window changes color from gray to yellow and displays different information.

Insert Commands

Insert»Displayable Object

Security Level: 9
Shortcut Keys: <Insert>

The **Insert»Displayable Object...** command invokes a list box you use to select a pre-existing object for display on the active control panel. This command does not create a new object, it inserts on the screen a display element representing the selected object.

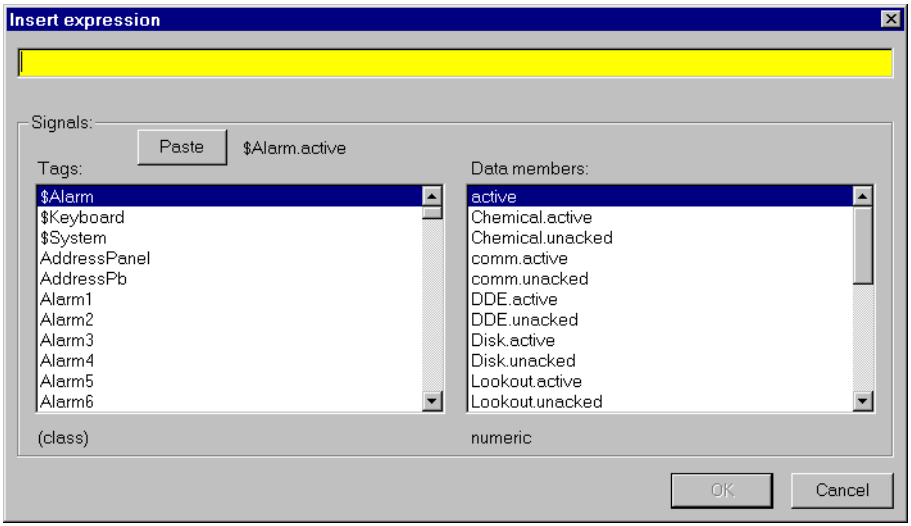


☞ **Note** ***Immediately after clicking on the* OK *button, a second dialog box appears for you to use in choosing the display characteristics of the selected object. What this dialog box looks like depends on the object class of the selected tagname.***

Insert»Expression

Security Level: 9
Shortcut Keys: <CTRL+E>

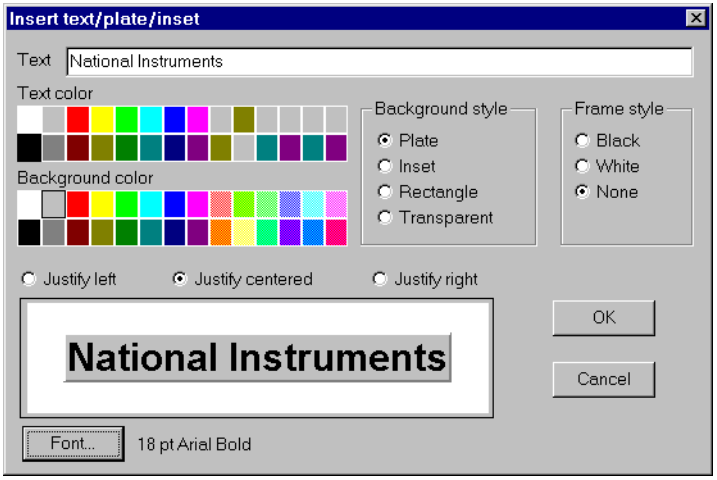
The **Insert»Expression...** command is a flexible, real-time calculator. It creates, calculates, and then displays the result of spreadsheet-style formulas that include a mixture of constants and signals from objects. There are over fifty built-in functions that you can use in expressions including logical, mathematical, statistical, text and trigonometric functions. See Chapter 7, *Expressions*, for more information.



Insert»Text/Plate/Inset

Security Level: 9
Shortcut Keys: <CTRL+T>

Use **Insert»Text/Plate/Inset...** to add text and/or three-dimensional plates and beveled insets to a control panel. This dialog box has two windows—one to enter the text you want (if any), and another larger window that serves as a preview screen for your current selections. The preview window lets you view the display characteristics before you select **OK** and insert the text, plate or inset.

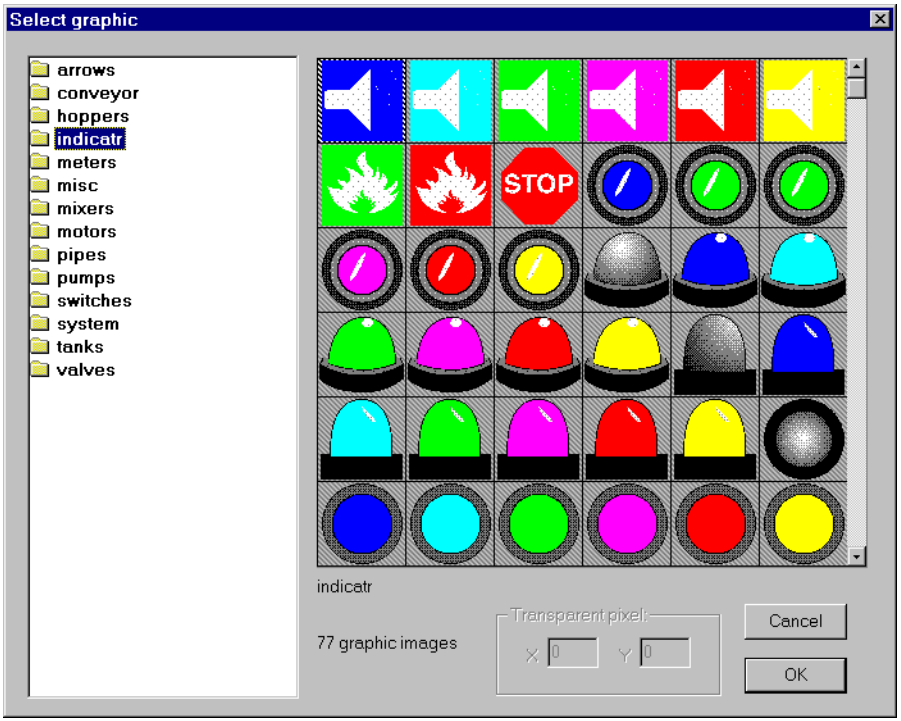


You can easily create plates and insets by selecting the appropriate **Background style** without entering any text. Plates and insets are useful tools that you can use generously when creating control panel layouts. They work especially well for grouping or segregating data. Take a few minutes to experiment with different fonts, text colors, background colors, and so on.

Insert»Graphic

Security Level: 9
Shortcut Keys: none

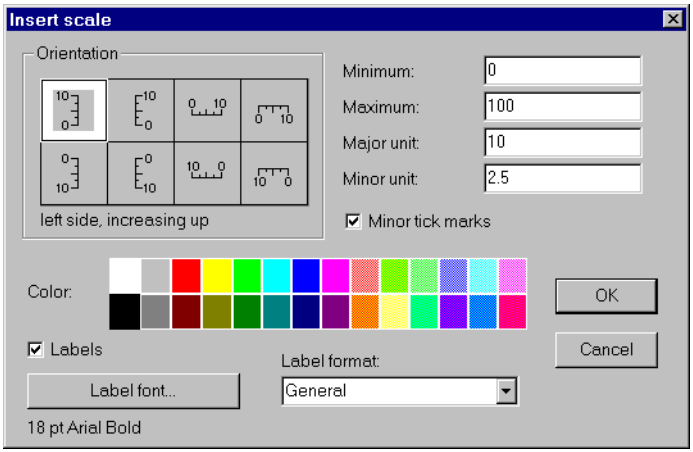
Use the **Insert»Graphic...** command to display Windows Device-Independent Bitmap (`.BMP`) and Windows Metafile (`.WMF`) graphic files on a control panel. This dialog box lists all bitmap and Windows metafiles located in the `GRAPHICS` subdirectories under the root `LOOKOUT` directory. See Chapter 8, *Graphics*, for more information.



Insert»Scale

Security Level: 9
Shortcut Keys: none

The **Insert»Scale...** command invokes a dialog box you use to create your own custom scale in a matter of seconds. Scales are useful with bar graphs, sliders and trend charts.

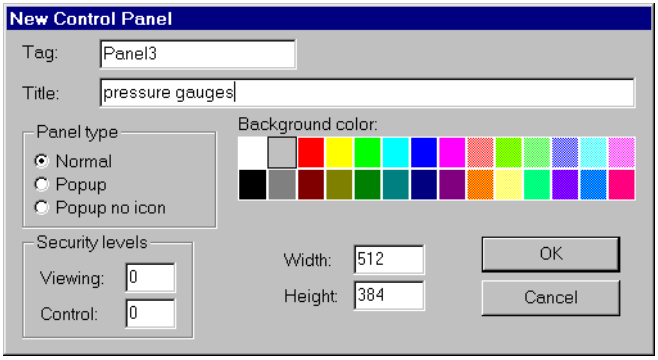


Click on the **Orientation** you want and fill in the **Minimum** and **Maximum** values for the scale. **Major unit** specifies the number of units between major tick marks. Only major units have labels. You can remove labels from your scale altogether by deselecting the **Labels** box. **Minor unit** specifies the number of units between minor tick marks. You can remove minor tick marks by deselecting the **Minor tick marks** box. **Color**, **Label format**, and **Label font** can be specified for special effects.

Insert»Control Panel

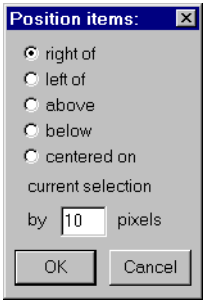
Security Level: 9
Shortcut Keys: <CTRL+P>

Use the **Insert»Control Panel...** command to add control panels to your process. Control panels are objects and, therefore, must have unique **Tag** names. However, you can repeat the **Title** of a control panel as many times as you want. Additionally, you do not have to follow the tagname syntax guidelines when filling out a panel **Title** field. See the *Panel* object in Chapter 18, *Object Class Reference*, for more information.



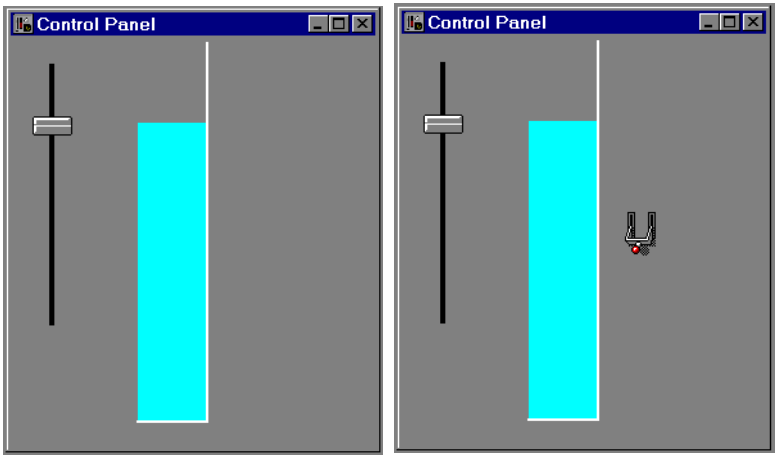
Insert»Positions

Security Level: 9
Shortcut Keys: none



The **Insert»Positions...** command tells Lookout where on a control panel to place the next inserted item. The direction and distance specified by this dialog box are relative to the currently selected item on the active control panel.

The following example demonstrates how the position settings would work in a typical case



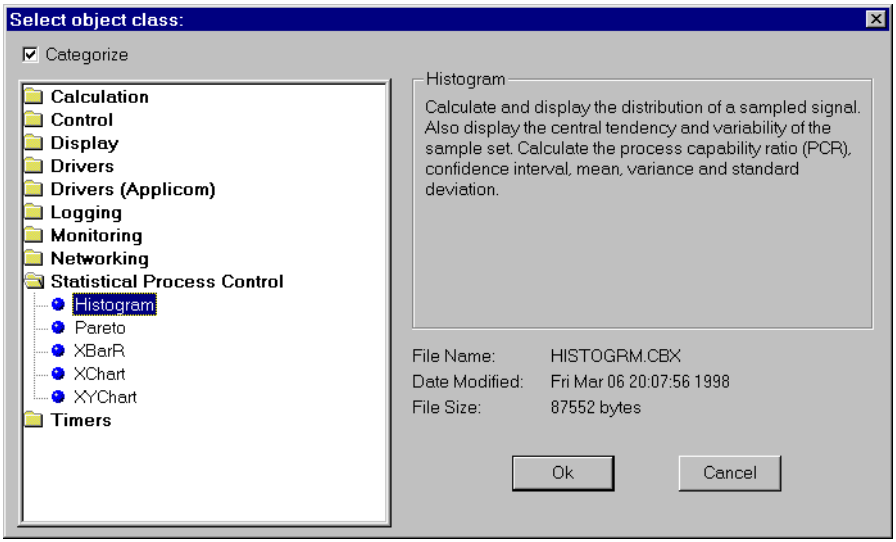
Select a bar graph

The next item is inserted 10 pixels to the right of the selected item.

Object Commands

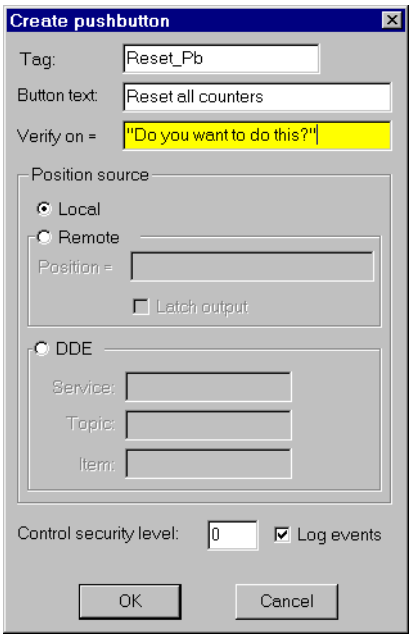
Object»Create

Security Level: 9
Shortcut Keys: <CTRL+Insert>



The **Object»Create...** command invokes the Select Object Class dialog box. This list box displays all the available object classes divided into categories. Objects in each category are listed in alphabetical order. You can select an object class with the mouse and select the **OK** button—or just double-click on the class. See Chapter 18, *Object Class Reference*, for more information on specific objects.

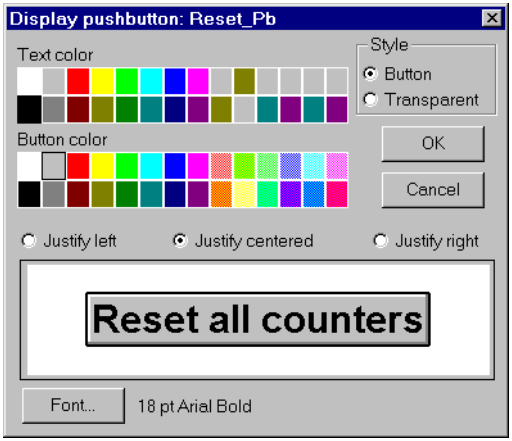
Deselecting the **Categorize** checkbox lists all Lookout objects alphabetically.



After choosing an object class, Lookout displays a dialog box that you use to define the object parameters. Fill in the parameters according to your requirements and click on **OK**. See Chapter 18, *Object Class Reference*, for more information on object classes.

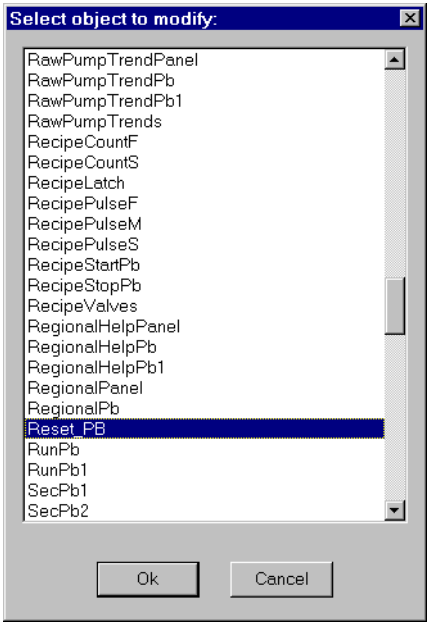
When you click on the **OK** button, Lookout creates a new object with the specified connections, if any. Lookout does not have to display objects on a control panel; they can exist without being displayed. However, many objects immediately prompt you to select display characteristics for insertion on a control panel (for example, the Pushbutton object).

If you select Cancel, Lookout does not display the object you just created. However, the object still exists. You can display the object at a later time using the **Insert»Displayable Object...** command.



Object»Modify

Security Level: 9
Shortcut Keys: none

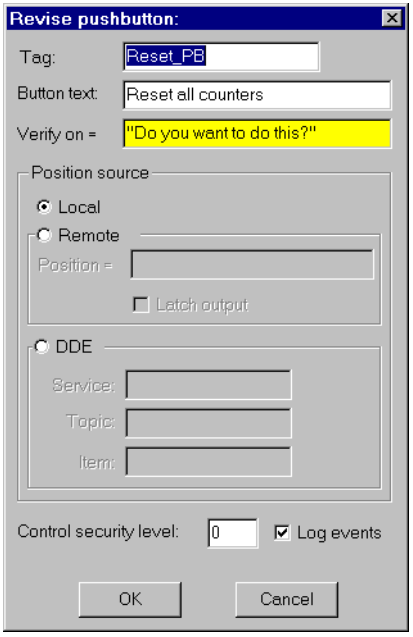


Use the **Object»Modify...** command to edit any pre-existing object, no matter what class the object belongs to. This command invokes a dialog box with an alphabetical listing of all existing tagnames.

Choose a tagname and click on **OK**, or just double-click on the tagname.

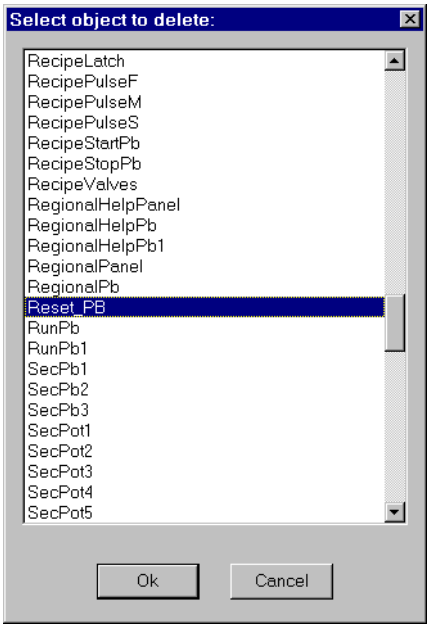
A dialog box for the identified object appears with the pre-existing parameters already filled in. You can modify the object by altering its parameters—even change the tagname. See *Configuration Shortcuts* in Chapter 3, *Getting Started*, for more information on modifying objects.

Note ***Changing the tagname does not create a second, new object. It replaces the existing tagname with a new one. Also, all other connected objects immediately reflect the new tagname.***



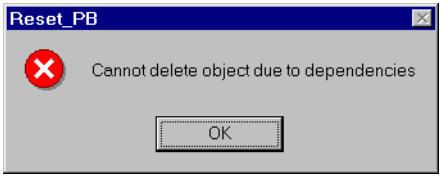
Object»Delete

Security Level: 9
Shortcut Keys: none



Use the **Object»Delete...** command to choose a specific object from an alphabetical list and delete it. The Select Object dialog box appears, prompting you to pick a tagname for deletion.

If you select an object that another object is using (that is, another object is connected to it), or that is displayed on a control panel, Lookout refuses to permit the deletion and gives a warning. You must first delete all references to, and displays of, the object you want to delete.



Objects are not the only things that can be dependent on other objects. Expressions that use the tagname of an object you are trying to delete also create a dependency.

☞ **Note** ***If you think you have deleted all the dependencies of an object and still get this message box, you may want to look at the source file for your process (*** .LKS ***file). You can then do a search command with your text editor or word processor to find all occurrences of the problem tagname.***

Object»Edit Connections

Security Level: 9
Shortcut Keys: none

Use the **Object»Edit Connections...** command to tie signals to writable data members of an object. You most commonly use this command to connect signals to outputs on controllers and other field devices (PLCs, RTUs, Loop Controllers, and so on). Other object classes such as Pots and Panels also take advantage of this function. See Chapter 5, *Developer Tour*, for more information on connecting objects.

When you select **Object»Edit Connections...**, notice that many tagnames are not listed. Lookout displays only the objects that accept signals. See Chapter 5, *Developer Tour*, for more information.

Object»Edit Database

Security Level: 9
Shortcut Keys: none

Use the **Object»Edit Database...** command to define parameters of the database members for an object. These include alias names, descriptions, alarm parameters, scaling parameters, logging parameters, and so on. You can also use this command to import and export databases. You most commonly use this command in association with DataTables and driver objects. See Chapter 5, *Developer Tour*, for more information on database editing.

Arrange Commands

The Arrange commands are useful tools for quickly building professional looking control panels. Understanding and mastering these commands significantly speeds up development time and improves the look and feel of your screens. Although this manual explains the functionality of each command, the best way to familiarize yourself with the entire command set is to experiment with commands individually.

Arrange»Align

Security Level: 9
Shortcut Keys: none

The Align command organizes a group of selected objects on a control panel. It is often used in conjunction with the **Arrange»Space Evenly** command. First select the objects you want to align. Then select the appropriate align command (described in the following table).

Command	Description	Shortcut Key
Left	Aligns selected symbols to the left side of the bounding box	<CTRL+LEFT>
Right	Aligns selected symbols to the right side of the bounding box	<CTRL+RIGHT>
Top	Aligns selected symbols to the top of the bounding box	<CTRL+UP>
Bottom	Aligns selected symbols to the bottom of the bounding box	<CTRL+DOWN>
Vert. Center	Aligns selected symbols to the vertical center of the bounding box	<F9>
Horiz. Center	Aligns selected symbols to the horizontal center of the bounding box	<SHIFT+F9>

Note ***Instead of selecting an entire group of objects by click-dragging, you can select multiple objects by holding the*** <SHIFT> ***key down while individually selecting each object. You can also use this technique to skip objects that you would otherwise select within the bounding box.***

Arrange»Space Evenly

Security Level: 9

This command is useful when trying to align and evenly space a number of objects on a control panel. Lookout evenly spaces the selected objects, on their centers, either across or down the bounding box. As mentioned above, this command is often used in conjunction with the Align command.

Command	Description	Shortcut Key
Across	Evenly spaces the selected objects horizontally across the bounding box	<ALT+RIGHT>
Down	Evenly spaces the selected objects horizontally down the bounding box	<ALT+DOWN>

Arrange»Group

Security Level: 9
Shortcut Keys: <CTRL+G>

The **Arrange»Group** command combines the selected individual objects, including text, into a single unit. This command is typically used to combine or group multiple related objects for further alignment. You can also create layered subgroups (that is, groups within larger groups).

Arrange»Ungroup

Security Level: 9
Shortcut Keys: <CTRL+U>

The **Arrange»Ungroup** command ungroups the currently selected group into its various components. This command works on only one group at a time. If you have layered subgroups, you can selectively ungroup them one layer at a time. Notice that if two separate groups are selected in a single lasso, the **Arrange»Ungroup** command is not available.

Arrange»Move to Front

Security Level: 9
Shortcut Keys: <CTRL+F>

Arrange»Move to Front moves the currently selected item(s) on top of all the other items, shifting all other displayed items back.

Arrange»Move to Back

Security Level: 9
Shortcut Keys: <CTRL+B>

Arrange»Move to Back moves the currently selected item(s) behind all the other items, shifting all other displayed items closer to the front.

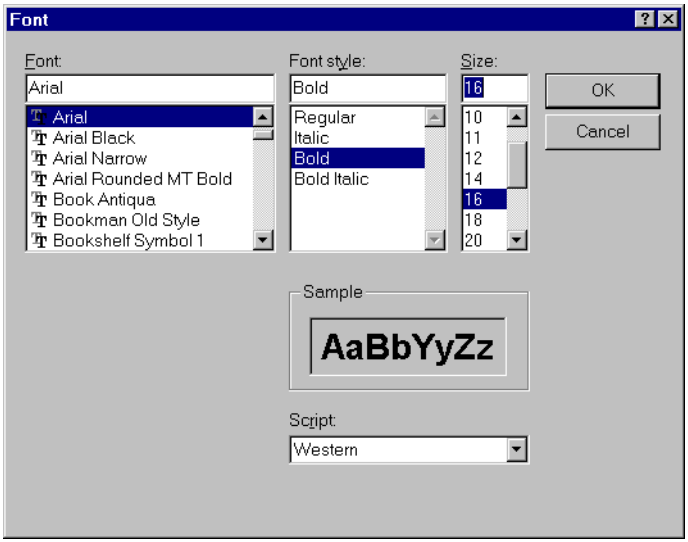
Change Functions

Use change functions to modify how text, bar charts, numeric and slider representations of pot objects and other display elements look on the screen. This is a useful tool for quickly making global changes to many items at once (for instance, to change the text color for all items currently selected). It is also the only way to change the background color of a slider.

Change»Font

Security Level: 9
Shortcut Keys: none

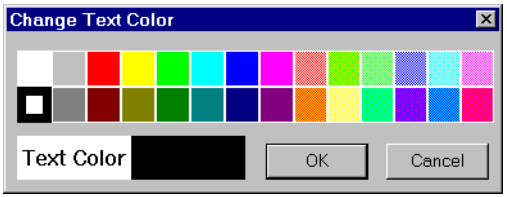
Change»Font... invokes a dialog box that modifies the font on all the currently selected items.



Change»Text Color

Security Level: 9
Shortcut Keys: none

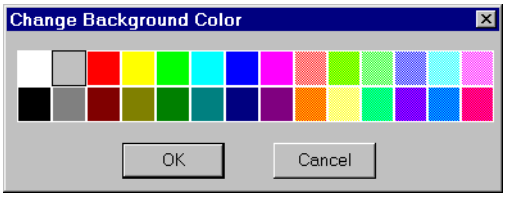
Change»Text Color... invokes a dialog box that you can use to modify the text color of all the currently selected items. This command also affects the bar (foreground) color on bar graphs.



Change»Background Color

Security Level: 9
Shortcut Keys: none

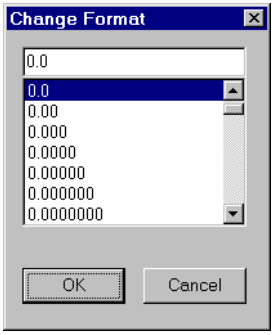
Change»Background Color... invokes a dialog box that modifies of the background color on all the currently selected items.



Change»Numeric Format

Security Level: 9
Shortcut Keys: none

Change»Numeric Format... invokes a dialog box you can use to change the numeric format on all the currently selected items. See *Numeric Data Members* in Chapter 2, *Introduction*, for more information on numeric formats.



Change»Justify Text

Security Level: 9
Shortcut Keys: none

The **Change»Justify Text** subcommands modify text alignment on all the currently selected text items. This command also applies to digitally displayed data.

Command	Description	Shortcut Key
Left	Aligns selected text and data to the left side	none
Right	Aligns selected text and data to the right side	none
Center	Centers selected text and data	none

Part III

Object Reference

Chapter 18, *Object Class Reference*, describes Lookout object classes, listed in alphabetical order. Input parameter syntax and data members are documented for each object class, along with a description of the functionality of each object class and an example of how to use the object.

Both the Lookout native objects and the various driver objects are detailed in this section.

© National Instruments Corporation

18

Object Class Reference

This chapter describes Lookout object classes, listed in alphabetical order. Input parameter syntax and data members are documented for each object class, along with a description of the functionality of each object class.

☞ **Note** ***Lookout assists you in building graphical screen displays when possible. Immediately after creating a new object that supports implicit data members or is displayable, Lookout automatically invokes another dialog box. If the object is displayable (like Pots, Switches, and Trends), the appropriate* Display Parameter *dialog box appears, prompting you to select a display type. If the object is not displayable but supports an implicit signal (like Counters, LatchGates and OneShots), Lookout automatically invokes the* Insert Expression *dialog box and pastes the appropriate tagname in the window. You can then click on the* OK *button to display the result of the signal as an expression on a control panel.***

You can elect to not display an object or its signal by clicking on the **Cancel** button at any time. If you change your mind later, you can display the object or its signal using the **Insert»Displayable Object…** or **Insert»Expression…** commands, respectively.

Some object classes have neither a display member nor an implicit signal—instead they have multiple data members. In these cases, Lookout does not automatically invoke a second dialog box. However, if you want to display the result of a logical or numeric data members on a control panel, you can use the **Insert»Expression…** command and choose the appropriate tagname and data member.

See Chapter 5, *Developer Tour*, for more information on creating objects, modifying their databases, and linking them together.

AB_PLC2, AB_PLC5, AB_SLC500

Lookout uses the AB object classes to communicate with the Allen-Bradley family of PLC controllers using a variety of interfaces.

Lookout can communicate with a member of the PLC-2 family in the following ways:

- Through a Data Highway Plus (DH+) connection to an Allen-Bradley 1785-KA3 PLC-2 adapter module using an Allen-Bradley 1784-KT, 1784-KTx, or 1784-PCMK card, or an S-S Technologies 5136-SD direct-link interface card installed in the computer,
- Through the serial port using an Allen-Bradley KF2 module (which converts serial DF1 to DH+) to an Allen-Bradley 1785-KA3 PLC-2 adapter module, or
- Through a direct DF1 serial connection to the PLC programming port.

Lookout can communicate with a member of the PLC-5 family in the following ways:

- Through a direct Ethernet connection to the PLC AUI port,
- Through a direct DH+ connection using a 1784-KT, 1784-KTx, 1784-PCMK, or 5136-SD card installed in the computer,
- Through the serial port via a KF2 module which converts serial DF1 to DH+, or
- Through a direct DF1 serial connection to the PLC programming port.

Lookout can communicate with a member of the SLC-500 family in the following ways:

- Through a direct DH+ connection using a 1784-KT, 1784-KTx, 1784-PCMK, or 5136-SD card installed in the computer,
- Through the serial port via a KF2 module which converts serial DF1 to DH+,
- Through the serial port using an Allen-Bradley 1747-KE card (which plugs into the SLC chassis and converts DF1 to DH 485),
- Through the serial port using a stand-alone Allen-Bradley 1770-KF3 communication interface module which converts DF1 to DH 485,

- Through a direct DF1 serial connection to a SLC 5/03 or SLC 5/04 programming port, or
- Through a direct DH485 connection using a 1784-PCMK card in conjunction with either a 1747-AIC or 1761-NET-AIC module.

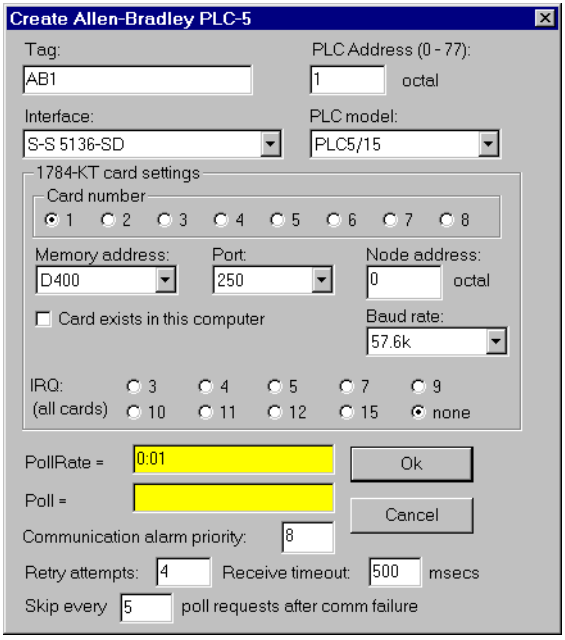


Figure 18-1. Allen-Bradley Parameter Dialog Box

PLC Address refers to the PLC network node address setting as configured on the physical device. If devices share a common **Interface**, they require unique addresses. When using DF1 protocol (serial communications), valid addresses range from 0 to 254 decimal. When using DH+, valid addresses range from 0 to 77 octal.

PLC Model specifies the particular type of PLC or SLC you are representing with this object. The **PLC Model** you select determines what native data members comprise the object.

PollRate is a numeric expression that determines how often to poll the device. Lookout polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *Introduction*, for information on entering time constants.

Poll is a logical expression. When this expression changes from false to true, Lookout polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Communication alarm priority determines the priority level of the alarms generated by the AB object.

Retry attempts specifies the consecutive number of times Lookout attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the object generates an alarm and begins to **Skip every *n* poll requests after comm failure**. Once Lookout reestablishes communications, it polls the device on its regular cycle, as defined by **PollRate**.

Receive timeout is the time delay Lookout waits for a response from a device before retrying the poll request.

Interface identifies the type of communication hardware you are using. The selection you make here determines what protocol parameters you have to specify. The paragraphs that follow describe interface-specific protocol parameters.

Allen-Bradley Serial Port Interface Parameters

The `KE/KF/Serial` **Interface** selection enables serial port communication via a KE card, a KF3 module or KF2 module. When using your serial port, Lookout employs the Allen-Bradley full-duplex (peer-to-peer) DF1 protocol. The previous diagram shows an Allen Bradley object configured for serial communications.

Serial port specifies which RS-232C port the object uses for communication to the physical device.

Data rate, **Parity**, and **Error detection** reference the settings on the hardware device. The AB object classes support both *BCC* (block check character) and *CRC* (cyclic redundancy check) error detection. BCC provides a medium level of data security. CRC ensures a higher level of data security. Choose the settings as configured on your PLC or SLC.

Phone number specifies the number to be dialed if the serial port setting is configured for dial-up. This number only applies to the individual object. See *Options*»*Serial Ports* of Chapter 17, *Edit Mode Menu Commands,* for more information.

Allen-Bradley DH+ Interface Parameters

Note ***When you configure an AB_PLC2, AB_PLC5, or AB_SLC500 for DH+, Lookout creates a file called*** `ALLBRAD.INI`***. This file contains the configuration settings that you enter for your KT card(s). If you plan to run the process file on a Lookout Runtime System, be sure to copy the INI file along with your process (*** `.LKP`***) file to the target computer.***

The `1784-KT`, `1784-KTx`, `1784-PCMK`, and `S-S 5136-SD` **Interface** selections enable direct connection of your computer to a DH+ network. The diagram below shows an AB_SLC500 configured for DH+ communications using a 1784-KTx card.

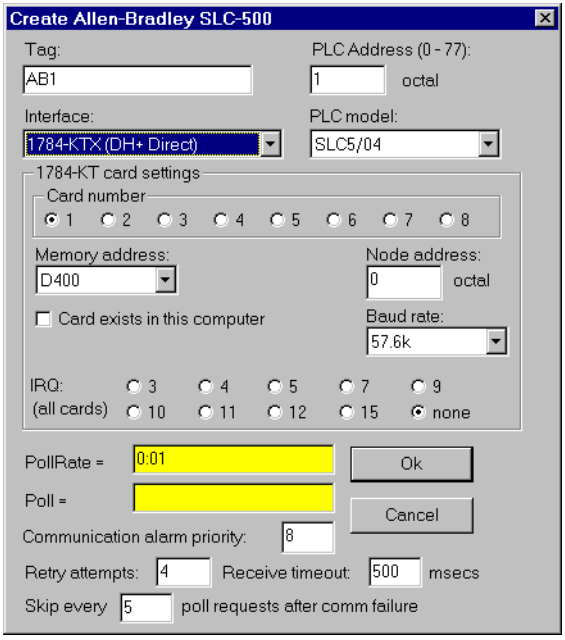


Figure 18-2. AB_SLC-500 Definition Parameters Dialog Box Configured for DH+ Communications

Card number selects which network interface card that the PLC is connected to in case your computer has multiple KT or S-S cards.

Card DH+ node address identifies the address of the interface card in the DH+ network. Valid addresses range from 0 to 77 octal. The node address of the card must be unique—that is, it must not be the same as the address of any other device on the DH+ network.

Memory address specifies the base address location of the selected interface card memory. Your selection should match the settings on the card. If you are using multiple interface cards, be sure each card has a unique address. The network interface cards use dual-ported memory. For this reason, if you are using a memory manager such as EMM386 it is important to add a memory exclusion statement to your `CONFIG.SYS` file. The table on the following page lists base memory address selections and corresponding exclusions for all legal memory addresses for the 1784-KT card.

The 1784-KT interface card has on-board network termination resistors. If you are using such a card and if your computer is the last node on the network and if the cable does not already have a terminating resistor on it, then select the **Enable link termination resistor** check box.

Use the **Card exists in this computer** check box to instruct Lookout whether or not to look for the interface card in the computer. *Be sure to check this box when you are ready to start polling your PLCs.* When you check this box and select **OK**, Lookout initializes the card, activates its self-test, and downloads its driver firmware. Then polling begins. Leave the **Card exists in this computer** check box deselected (this is the default setting) if the card is not in your computer (for example, if you are developing a process on a computer different from the one that will be running the process) or if you do not want to poll any PLC connected to the card.

If you deselect the **Card exists in this computer** check box, you are disabling communications using this interface card with all PLCs connected to it.

Table 18-1. Allen-Bradley DH+ Interface Memory Addresses

Memory Address	AB 1784-KT Exclude	AB 1784-KTx 1784-PCMK Exclude	SS-5136-SD Exclude*	Recommendation
A000	A300-A3FF	A000-A0FF	A000-A3FF	Typically used by VGA drivers. Use if no other option is available.
A400	A700-A7FF	A400-A4FF	A400-A7FF	
A800	AB00-ABFF	A800-A8FF	A800-ABFF	
AC00	AF00-AFFF	AC00-ACFF	AC00-AFFF	

Table 18-1. Allen-Bradley DH+ Interface Memory Addresses (Continued)

Memory Address	AB 1784-KT Exclude	AB 1784-KTx 1784-PCMK Exclude	SS-5136-SD Exclude*	Recommendation
B000	B300-B3FF	B000-B0FF	B000-B3FF	Used by MDA & CGA drivers. Use if no Dxxx option is available.
B400	B700-B7FF	B400-B4FF	B400-B7FF	
B800	BB00-BBFF	B800-B8FF	B800-BBFF	
BC00	BF00-BFFF	BC00-BCFF	BC00-BFFF	
C000	C300-C3FF	C000-C0FF	C000-C3FF	Typically used by BIOS. Use if no Dxxx option is available.
C400	C700-C7FF	C400-C4FF	C400-C7FF	
C800	CB00-CBFF	C800-C8FF	C800-CBFF	
CC00	CF00-CFFF	CC00-CCFF	CC00-CFFF	
D000	D300-D3FF	D000-D0FF	D000-D3FF	Normally available.Try to use one of these first.
D400	D700-D7FF	D400-D4FF	D400-D7FF	
D800	DB00-DBFF	D800-D8FF	D800-DBFF	
DC00	DF00-DFFF	DC00-DCFF	DC00-DFFF	

* The 5136-SD memory exclusions recommended above are based on 16K memory mapping. Because you are using the SS card to emulate the KT card, there is no advantage to using its 32K memory access capability.

Baud rate (1784-KTx, 1784-PCMK, 5136-SD only) selects the baud rate of the DH+ network. The default is 57.6k baud. Before selecting a higher baud rate, be aware of that only a few PLCs (such as the SLC5/04) support higher baud rates; that every node on a DH+ network must support the baud rate used on that network; that the maximum network cable length is smaller for higher baud rates; and that the correct values for the termination resistors at the ends of the network cable are different for higher baud rates. Consult the manuals that came with your hardware for more detailed information.

IRQ (all cards) identifies the interrupt setting of all DH+ interface cards installed in the computer. This selection should match the IRQ settings on *all* of the interface cards.

Assigning an interrupt to the interface card(s) improves overall computer performance somewhat. Any time one of the cards receives an input, it generates an interrupt recognized by Lookout.

Caution ***Be sure to verify that no other drivers or cards are mapped to the selected memory address or use the same interrupt.***

Allen-Bradley Ethernet Interface Parameters

The `Ethernet` **Interface** selection enables direct communication between your computer and a PLC using a standard Ethernet network. The following diagram shows an AB_PLC5 configured for Ethernet communications.

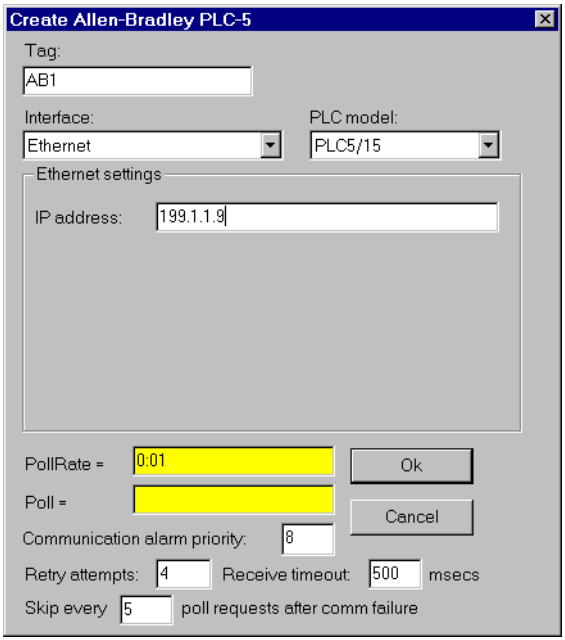


Figure 18-3. AB_PLC5 Definition Parameters Dialog Box Configured for Ethernet Communications

IP address specifies the Internet protocol address of the PLC. An Internet protocol address consists of four numbers, separated by periods. Each number ranges from zero to 255 decimal. Thus, a typical Internet address might be 128.7.9.231. Ensure that the **IP address** you enter matches the Internet protocol address of the PLC your object represents. You can also enter the IP address by name.

Using the 5136-SD card from S-S Technologies, Inc.

To use the 5136-SD card, select the S-S 5136-SD interface in the **Create Object** dialog box. It is not necessary to run the `sdinst.exe` program that ships with the card because Lookout downloads the KT-emulation module automatically as part of the initialization process. It is, however, necessary to tell Lookout the port address specified by the switch settings on the card.

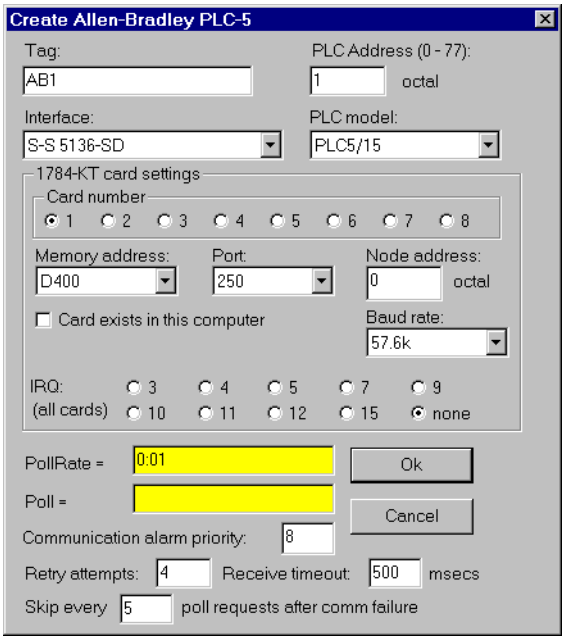


Figure 18-4. AB_PLC5 Definition Parameters Dialog Box Configured for the 5136-SD card

Allen-Bradley Data Members

Each AB object contains a great deal of data. All readable and writable members (inputs/outputs) are bundled with the object. As soon as you create an AB object you immediately have access to all the object data members.

The AB object classes automatically generate an efficient read/write blocking scheme based on the inputs and outputs you are using in your process file. You are not required to build your own I/O blocking table. However, you can ensure peak performance by organizing your PLC data into contiguous groups.

Table 18-2. AB_PLC2 Data Members

Data Member	Type	Read	Write	Description
0 - 7777	numeric	yes	yes	16-bit signed binary word ranging from –32,768 to +32,767
0_0 - 7777_17	logical	yes	no	One bit within a 16-bit binary word
CommFail	logical	yes	no	Object-generated signal that is ON if, for any reason, Lookout cannot communicate with the PLC.
OffHook	logical	no	yes	When true, instructs the PLC to retain exclusive use of its assigned communication port. This prevents Lookout from hanging up between polls, saving the redial overhead. This also prevents other blocks from communicating over the same channel.
Poll	logical	no	yes	When transitioned from false to true, the Lookout object polls the PLC device
PollRate	numeric	no	yes	Specifies the frequency at which the Lookout object polls the PLC device
Update	logical	yes	no	Object-generated signal that pulses each time the object polls the device

Table 18-3. AB_SLC500 Data Members

Data Member	Type	Read	Write	Description
B:0 - B255:255	numeric	yes	yes	16-bit signed binary word ranging from –32,768 to +32,767
B:0_0-B255:255_15	logical	yes	yes	One bit within a 16-bit binary word
B_0 - B255_4095	logical	yes	yes	One bit within the specified datafile. For example, B3_32 specifies datafile 3, word 2, bit 1.
C:0.ACC - C255:255.ACC	numeric	yes	yes	Counter accumulated value. Two-byte signed word ranging from –32,768 to +32,767

Table 18-3. AB_SLC500 Data Members (Continued)

Data Member	Type	Read	Write	Description
C:0.PRE - C255:255.PRE	numeric	yes	yes	Preset counter value. Two-byte signed word ranging from –32,768 to +32,767
C:CD -C255:CD	logical	yes	yes	Counter down-enable bit.
C:0_CU - C255:255_CU	logical	yes	yes	Counter up-enable bit.
C:0_DN -C:255:255_DN	logical	yes	yes	Counter done bit.
C:0_OV - C255:255_OV	logical	yes	yes	Counter overflow bit.
C:0_UA - C255:255_UA	logical	yes	yes	Counter update accumulation bit (HSC in fixed controller only)
C:0_UN - C255:255_UN	logical	yes	yes	Counter underflow bit.
CommFail	logical	yes	no	Object-generated signal that is ON if, for any reason, Lookout cannot communicate with the SLC.
F:0-F255:255	numeric	yes	yes	Floating point value
I:1 - I:30	numeric	yes	yes	Unsigned 16-bit input value ranging from 0 to 65,535
I:1_0 - I:30_15	logical	yes	yes	One bit within a 16-bit input word
N:0 - N255:255	numeric	yes	yes	16-bit signed integer value ranging from –32,768 to +32,767.
N:0_0 - N255:255_15	logical	yes	yes	One bit within a 16-bit signed integer word
O:1 - O:30	numeric	yes	yes	Unsigned 16-bit output value ranging from 0 to 65,535
O:1_0 - O:30_15	logical	yes	yes	One bit within a 16-bit output word
OffHook	logical	no	yes	When true, instructs the PLC to retain exclusive use of its assigned communication port. This prevents Lookout from hanging up between polls, saving the redial overhead. This also prevents other blocks from communicating over the same channel.

Table 18-3. AB_SLC500 Data Members (Continued)

Data Member	Type	Read	Write	Description
Poll	logical	no	yes	When transitioned from false to true, the Lookout object polls the SLC device
PollRate	numeric	no	yes	Specifies the frequency at which the Lookout object polls the SLC device
R:0.LEN - R255:255.LEN	numeric	yes	yes	Control “length” signed integer ranging from –32,768 to +32,767
R:0.POS - R255:255.POS	numeric	yes	yes	Control “position” signed integer ranging from –32,768 to +32,767
R:0_DN - R255:255_DN	logical	yes	yes	Control “done” single-bit logical indicator
R:0_EM - R255:255_EM	logical	yes	yes	Control “empty” single-bit logical indicator
R:0_EN -R255:255_EN	logical	yes	yes	Control “enable” single-bit logical indicator
R:0_ER - R255:255.ER	logical	yes	yes	Control “error” single-bit logical indicator
R:0_EU - R255:255_EU	logical	yes	yes	Control “enable unloading” single-bit
R:0-UL - R255:255_FD	logical	yes	yes	Control “found” single-bit logical indicator
R:0_IN - R255:255.IN	logical	yes	yes	Control “inhibit comparison” flag single-bit logical indicator
R:0_UL - R255:255_UL	logical	yes	yes	Control “unload” single-bit logical indicator
S:0 - S:96	numeric	yes	yes	SLC status file containing a signed integer ranging from –32,768 to +32,767 (see Allen-Bradley documentation)
S:0_0 -S:96_15	logical	yes	yes	Individual SLC status bits (see Allen-Bradley documentation)
T:0.ACC - T255:255.ACC	numeric	yes	yes	Accumulated timer value ranging from –32,768 to +32,767

Table 18-3. AB_SLC500 Data Members (Continued)

Data Member	Type	Read	Write	Description
T:0.PRE - T255:255.PRE	numeric	yes	yes	Preset timer value ranging from –32,768 to +32,767
T:0_DN - T255:255_DN	logical	yes	yes	Timer “done” single-bit logical indicator
T:0_EN - T255:255_EN	logical	yes	yes	Timer “enabled” single-bit logical indicator
T:0_TT - T255:255_TT	logical	yes	yes	Timer “timing” single-bit logical indicator
Update	logical	yes	no	Object-generated signal that pulses each time the object polls the device

Table 18-4. AB_PLC5 Data Members

Data Member	Type	Read	Write	Description
B:0 - B999:999	numeric	yes	yes	16-bit signed binary word ranging from –32,768 to +32,767
B_0 - For B999_15999	logical	yes	yes	One bit within the specified datafile; for example, B3_32 specifies datafile 3, word 2, bit 1.
C:0.ACC - C999:999.ACC	numeric	yes	yes	Counter accumulated value. Two-byte signed word ranging from –32,768 to +32,767
C:0.PRE - C999:999.PRE	numeric	yes	yes	Preset counter value ranging from –32,768 to +32,767
C:0_CD - C999:999_CD	logical	yes	yes	Counter down-enable bit.
C:0_CU - C999:999_CU	logical	yes	yes	Counter up-enable bit.
C:0_DN - C:999:999_DN	logical	yes	yes	Counter done bit.
C:0_OV - C999:999_OV	logical	yes	yes	Counter overflow bit.
C:0_UA - C999:999_UA	logical	yes	yes	Counter update accumulation bit (HSC in fixed controller only)
C:0_UN - C999:999_UN	logical	yes	yes	Counter underflow bit.

Table 18-4. AB_PLC5 Data Members (Continued)

Data Member	Type	Read	Write	Description
CommFail	logical	yes	no	Object-generated signal that is ON if, for whatever reason, Lookout cannot communicate with the PLC.
F:0-F999:999	numeric	yes	yes	Floating point value
I:0 - I:277	numeric	yes	yes	Unsigned 16-bit input value ranging from 0 to 65,535
I:0_0 - I:277_17	logical	yes	yes	One bit within a 16-bit input word (octal)
N:0 - N999:999	numeric	yes	yes	16-bit signed integer value ranging from –32,768 to +32,767.
N:0_0 - N999:999_15	logical	yes	yes	One bit within a 16-bit signed integer word
O:0 - O:277	numeric	yes	yes	Unsigned 16-bit output value ranging from0 to 65,535
O:0_0 - O:277_17	logical	yes	yes	One bit within a 16-bit output word (octal)
OffHook	logical	no	yes	When true, instructs the PLC to retain exclusive use of its assigned communication port. This prevents Lookout from hanging up between polls, saving the redial overhead. This also prevents other blocks from communicating over the same channel.
Poll	logical	no	yes	When transitioned from false to true, the Lookout object polls the PLC device
PollRate	numeric	no	yes	Specifies the frequency at which the Lookout object polls the PLC device
R:0.LEN - R999:999.LEN	numeric	yes	yes	Control “length” signed integer ranging from –32,768 to +32,767
R:0.POS - R999:999.POS	numeric	yes	yes	Control “position” signed integer ranging from –32,768 to +32,767
R:0_DN - R999:999_DN	logical	yes	yes	Control “done” single-bit logical indicator

Table 18-4. AB_PLC5 Data Members (Continued)

Data Member	Type	Read	Write	Description
R:0_EM - R999:999_EM	logical	yes	yes	Control “empty” single-bit logical indicator
R:0_EN - R999:999_EN	logical	yes	yes	Control “enable” single-bit logical indicator
R:0_ER - R999:999.ER	logical	yes	yes	Control “error” single-bit logical indicator
R:0_EU - R999:999_EU	logical	yes	yes	Control “enable unloading” single-bit logical indicator
R:0_FD - R999:999_FD	logical	yes	yes	Control “found” single-bit logical indicator
R:0_IN - R999:999.IN	logical	yes	yes	Control “inhibit comparison” flag logical indicator
R:0_UL - R999:999_UL	logical	yes	yes	Control “unload” single-bit logical indicator
S:0 - S:127	numeric	yes	yes	PLC status file containing a signed integer ranging from –32,768 to +32,767 (see Allen-Bradley documentation)
S:0_0 - S:127_15	logical	yes	yes	Individual PLC status bits (see Allen-Bradley documentation)
T:0.ACC - T999:999.ACC	numeric	yes	yes	Accumulated timer value ranging from –32,768 to +32,767
T:0.PRE - T999:999.PRE	numeric	yes	yes	Preset timer value ranging from –32,768 to +32,767
T:0_DN - T999:999_DN	logical	yes	yes	Timer “done” single-bit logical indicator
T:0_EN - T999:999_EN	logical	yes	yes	Timer “enabled” single-bit logical indicator

Table 18-4. AB_PLC5 Data Members (Continued)

Data Member	Type	Read	Write	Description
T:0_TT - T999:999_TT	logical	yes	yes	Timer “timing” single-bit logical indicator
Update	logical	yes	no	Object-generated signal that pulses each time the object polls the PLC device

Allen-Bradley Error Messages

AB objects report the statuses of commands they issue to AB devices. When Lookout receives a response from an AB device, it reads the status (STS) byte and, if necessary, the extended status (EXT STS) byte to verify the device executed the Lookout command properly. If the command was not executed properly, Lookout reports the failure as an alarm containing the status code and its meaning. The following is an example of such an alarm:

```
EXT STS = 0F: not enough levels in address
```

AB object classes can also generate alarms internally. Below is a list of AB alarms generated by Lookout, their descriptions, and possible responses. In the messages, *KT* is used to refer to any of the DH+ interface cards (1784-KT, 1784-KTx, 1784-PCMK, or 5136-SD) and *SS* is used to refer to the 5136-SD card.

Cannot resolve ip address: *address*
The AB object failed to find any node on the network that corresponds to the given IP address. Confirm that the IP address entered in the **Modify Object** dialog box is correct.

Cannot get session id from plc
The AB object sent a message to the PLC requesting a TCP/IP session and failed to receive a satisfactory response.

Cannot communicate with device (code=dd)
The AB object timed out while waiting for a response (via TCP/IP) from the PLC. If the code is 0, the object timed out while trying to establish the TCP/IP connection; if the code is 1, the object timed out while waiting for a session id from the PLC; if the code is 2, the object timed out while waiting for a response to a poll request. Confirm that the IP address of the PLC has been entered correctly and that the PLC is reachable over the TCP/IP network.

Download of file sdipds.ss1 failed
Lookout was unable to write the KT-emulation program file to the 5136-SD card physical memory. This could be due to either an invalid port or memory address or to a faulty or improperly seated card.

Invalid memory address for card: *0xAAAA*
The memory address specified for the card (for example, D400) is not valid for this model of card. The address must be a multiple of 0x0100 and lie in the range 0xA000 to 0xDF00. Moreover, the KT, KTx, PCMK, and 5136-SD cards support different sets of valid memory addresses. See the documentation that shipped with the card for details.

Invalid node address for card: *xx*
Node addresses must be between 0 and 63 decimal.

Invalid port number for SS card: *0xPPP*
The port number specified for the 5136-SD card is invalid. See the documentation that shipped with the card for the list of valid port addresses.

Invalid port or memory address
Lookout was unable to write to the 5136-SD card physical memory. This could be due to either an invalid port or memory address or to a faulty or improperly seated card.

KT card failed to find resources
KT card receive mailbox is in an invalid state
KT card send mailbox is in an invalid state
You will probably never see one of these alarms. If you do, call National Instruments and ask for Technical Support.

KT card dual-ported memory test failed at location *xxxx*
The interface card failed a memory test when it was first powered on. The memory test reads, writes and rereads the dual-ported memory to ensure memory access by the card. Verify that the card is configured for the memory address that you specified. Verify that your memory manager (like EMM386) excludes the appropriate portion of memory. Verify that your card is not trying to use the same memory location as another card. You may need to restart the card by calling up the AB object definition dialog box and selecting **OK**. If that does not work try rebooting the computer. Other causes can include memory conflicts, a bad interface card, or a misbehaving driver.

KT card CTC test timed out
KT card CTC test failed with status code *xx*
The interface card failed the Counter Timer Circuit test when it was first powered on. This test verifies proper functionality of the card timer and counter modes over all CTC channels. You may need to restart, reseat, or replace the interface card.

KT card timed out while loading protocol code
KT card failed with status code *xx* while loading protocol
Lookout was not able to transfer a loader file to the card and subsequently download the card protocol firmware. Try to restart the interface card by calling up the AB object definition dialog box and selecting **OK**. If that does not work try rebooting the computer.

KT card is no longer responding
The Lookout AB object did not receive the interface card heartbeat within the last second. Normally, the card generates a heartbeat any time it receives the DH+ network token. If the alarm does not deactivate after 30 seconds, try to restart the card by calling up the AB object definition dialog box and selecting **OK**. If that does not work try restarting Lookout or rebooting the computer.

KT card memory address conflicts with card *n*
Lookout found another interface card with the same memory address. Be sure that the memory address on each interface card is different and that the corresponding **Memory address** in the Lookout object matches the card address.

KT card not present in this computer
The **Card exists in this computer** check box is deselected (this is the default setting). Select **Object»Modify** to retrieve the PLC definition parameters dialog box and select the **Card exists in this computer** check box and **OK** to initialize the card.

KT card RAM test timed out
KT card RAM test failed with status code *xx*
The interface card failed a memory test when it was first powered on. The memory test writes a pattern to on-board RAM and reads its content to verify the card memory is working. Confirm that the memory address specified in the **Object»Modify** dialog box is correct. You may need to restart, reseat, or replace the card.

KT card signature test failed
The AB object does not recognize the card. Make sure that the interface card is actually installed in the PC and that you indicated the correct memory address in Lookout. You may need to reseat the card, or the card may require repair. Also ensure that you did not identify more **Card Numbers** than actual physical cards.

KT card SI0 test timed out
KT card SI0 test failed with status code *xx*
The interface card failed the Serial Input Output test when it was first powered on. You may need to restart, reseat, or replace the interface card.

KT card send mailbox timed out
The AB object timed out while waiting for the KT card to signal that it is ready to be given a new message to send. This is most likely due to a communication problem between the computer and the PLC. Confirm that the network cable is properly installed and that the PLC is turned on.

NAK response received
The AB object received a NAK (not acknowledged) response when it polled the device. The device received a command from Lookout but it did not accept the message. The command that the device received may be incomplete or contain irregularities due to poor network performance. If your Serial Port is configured for radio, you may need to adjust the **RTS delay off** setting. Also consider increasing the number of **Retry attempts**.

No response -- no ACK for our transmission
Lookout is not getting any response from the device. This could be caused by just about anything. Verify that the **Data rate**, **Parity**, and **Error detection** settings are the same as the settings on the device. Make sure you are using the proper **Serial port** on your computer. Verify that the device interface module and other network equipment is connected and working properly. If you are using a modem, verify that your object **Phone number** and the serial port **Dial-up** settings are correct. This may also be caused by low level noise or reflections on the highway, or marginal circuitry on a card.

No response within timeout period
No response received after receiving ENQ
The AB object received an acknowledgment of its poll from the device. The device accepted the command from Lookout. However, the device did not appear to send anything else back in response. You may have to increase **Receive timeout** to make sure Lookout allows enough time to receive the message.

EOT response received
The AB object received an EOT (end of transmission) response when it polled the device indicating that the device did not have a message ready to give in response to the Lookout poll request. It is unlikely that you will ever see this error message.

Received TSN does not match
Response message garbled -- bad CRC or bad BCC
Response message garbled -- no DLE EXT
Response message garbled -- bad DLE follower
The AB object is receiving messages from the device. However, the messages may be failing the selected data integrity test. Verify that the object **Error detection** setting is the same as the settings on the device. Another cause may be that the last part of the message is actually getting clipped off before it is completed. You may have to increase the **Receive gap** Serial Port setting to ensure Lookout is receiving the entire message. If your Serial Port is configured for radio this could be caused by an audible squelch tail occurring at the end of a radio transmission. You may need to adjust the **RTS delay off** or the **CTS timeout** settings. Also consider increasing the number of **Retry attempts**.

Socket communications error *dd*: *msg*
The AB object has encountered a problem while attempting to communicate using TCP/IP. The error number *dd* and corresponding error message *msg* give further information. Confirm that the IP address of the PLC has been entered correctly and that the PLC is reachable over the TCP/IP network.

SS card failed
This message is suffixed with an error message read from the card itself. You may need to contact the vendor of your card for technical support.

SS card failed while performing diagnostics
Lookout successfully wrote the KT-emulation program to the 5136-SD card, but the program failed to terminate. Try running the `sdinst.exe` program that ships with the 5136-SD card, using the CHK option to confirm that the card is working properly.

Unable to access physical memory at segment *0xAAAA*
Lookout was unable to access the memory at the given segment address. The memory may already be in use by the operating system or by another application. Either change the object memory address (which may involve changing switch settings on the card itself) or, if you are using a memory manager, make sure that it is excluding the correct portion of memory.

Unable to open port *0xPPP* for SS card
Lookout was unable to open the port number specified for the 5136-SD card. Make sure that you have specified the port that is selected by the jumper settings on the card. Make sure that port and the following two ports (for example, 250, 251, and 252) are not in use by any other devices in your computer.

Unexpected data response length
Unexpected response
Unhandled error
You will probably never see one of these alarms. If you do, call National Instruments and ask for Technical Support.

Accumulator

Accumulator is a totalizer. It samples the numeric **Input** any time the **Sample** value transitions from off to on, adding each new sample to the previous running total. It resets the totalized value to zero when **Reset** transitions from off to on.

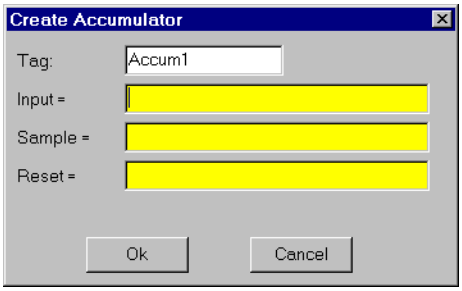
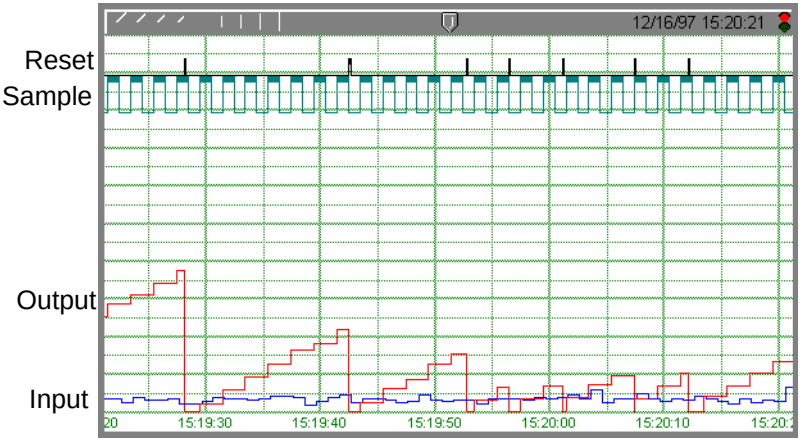


Figure 18-5. Accumulator Definition Parameters Dialog Box

Input is a numeric expression while **Sample** and **Reset** are logical expressions.

The following HyperTrend demonstrates the relationship between **Input**, **Sample**, and **Reset**.



☞ **Note** ***Accumulator does not have a display parameters dialog box. However, you can easily display the result of its output signal by referencing it in an expression.***

Accumulator Data Member

Table 18-5. Accumulator Data Member

Data Members	Type	Read	Write	Description
(implicit)	numeric	yes	no	Current totalized value, totalized since the most recent Reset signal. Updated at the defined Sample rate.

Comments **Reset *could be a regular pulse interval created by a Pulse timer, as shown in the dialog box on the previous page.***

The Counter object class totalizes a number of logical events and the Integral object class totalizes rates. Use Accumulator to totalize numeric variables.

Related Objects ***Average, Minimum, Maximum, Sample***

AdvantechPCL

The AdvantechPCL object class enables Lookout to communicate with the Advantech PC-MultiLab Card, model PCL-711B. The MultiLab Card installs in your computer and provides one analog output, eight analog inputs, 16 digital inputs and 16 digital outputs.

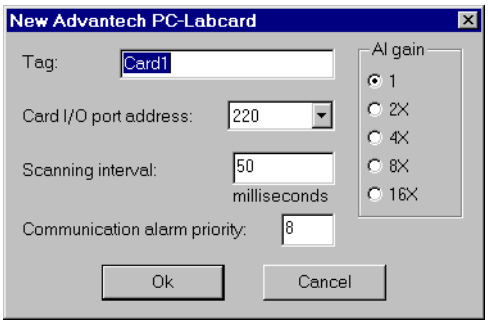


Figure 18-6. AdvantechPCL Definition Parameters Dialog Box

Card I/O port address indicates the base I/O port address you chose using DIP switch settings on the card. The Advantech card uses 16 consecutive I/O port address locations in your computer. So if you identify base address 220 (hex), it uses addresses 220 to 22F. Valid base addresses range from 000 to 3F0. If your computer does not have an Advantech card installed, be sure to select `no card`.

☞ **Note** ***I/O ports identify to the CPU the location of basic system components such as serial ports, video cards, and disk controllers. Check your computer hardware manual and add-on board configurations to be sure you are assigning a unique I/O port address to the Advantech card.***

Scanning interval is a numeric parameter that determines how often Lookout scans the card for changed values. Valid intervals range from 10 to 1000 milliseconds.

AI gain specifies the input amplification gains of all analog input signals (AI0 – AI7). For the highest possible resolution, you should amplify the input signals so that their maximum voltage swing equals their maximum input range. The following table lists recommended settings.

If your maximum voltage range is…	use A1 gain…
±5V	1
±2.5V	2X
±1.25V	4X
±0.625V	8X
±0.3125V	16X

Communication alarm priority determines the priority level of the alarms generated by the AdvantechPCL object class.

AdvantechPCL Data Members

The table that follows lists data members supported by the AdvantechPCL object class.

Data Member	Type	Read	Write	Description
AI0 – AI7	Analog	yes	no	Analog input where AI0 represents A/D 0 (pin 1 on CN1) and AI7 represents A/D 7 (pin 15 on CN1). Each input is a 12-bit integer, ranging from 0 to 4095.
AO0	Analog	no	yes	Analog output where AO0 represents D/A (pin 17 on CN1). This output is a 12-bit integer, ranging from 0 – 4095 (+5V or +10V, depending on jumper JP1).
CommFail	logical	yes	no	Object-generated signal that is on if, for whatever reason, Lookout cannot communicate with the card.
DI0 – DI15	logical	yes	no	Digital input where DI0 represents pin 1 and DI15 represents pin 16 on CN4. Returns False when a monitored signal is open (>2.0V), True when the signal is shorted (<0.8V).

Data Member	Type	Read	Write	Description
DO0 – DO15	logical	no	yes	Digital output where DO0 represents pin 1 and DO15 represents pin 16 on CN3.
Update	logical	yes	no	Object-generated signal that pulses each time the object scans the device.

Alarm

Alarm is a flexible and powerful object class you use to create various logical- and numeric-triggered alarms that are displayed in the alarm window.

☞ **Note** ***You should first read about the Lookout alarm system in Chapter 9,*** *Alarms****, to aid you in designing the most efficient alarming scheme for your application.***

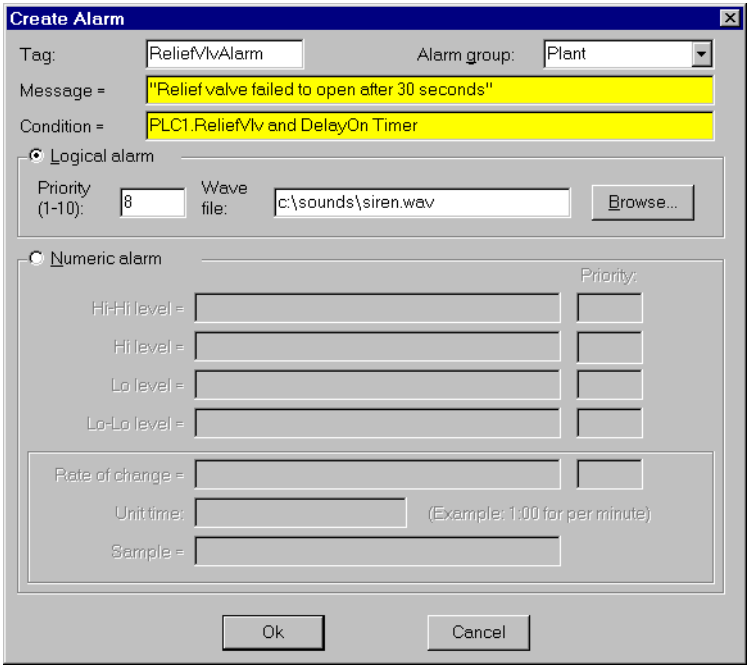


Figure 18-7. Typical Settings for a Logical Style Alarm

There are two basic alarm types: logical and Numeric. **When Logical alarm** is selected, Lookout prompts you to enter a logical expression in the **Condition** field. It then uses the logical **Condition** to trigger and reset the alarm. If the alarm **Condition** is true, the alarm is active, and if the condition is false, the alarm goes inactive. You can also connect an audio **Wave file** to individual logical alarms. See *Playwave* for additional information.

Lookout queues alarm `.WAV` files, with up to 100 files in the queue. Each alarm `.WAV` file plays completely before the next file plays. If more than

100 alarms fill the queue, new alarms cancel the currently playing files and begin playing instead.

When you pick the **Numeric** alarm selection, the name of the Condition field changes to **Signal**. This prompts you to enter a numeric expression in the field against which your various alarm setpoints are measured. **Hi-Hi**, **Hi**, **Lo**, and **Lo-Lo** are all numeric expressions. **Rate of change** generates an alarm when the signal is actively changing by the set amount for the period of time between any two **Sample** pulses. The **Unit time** setting determines the time units for the rate of change.

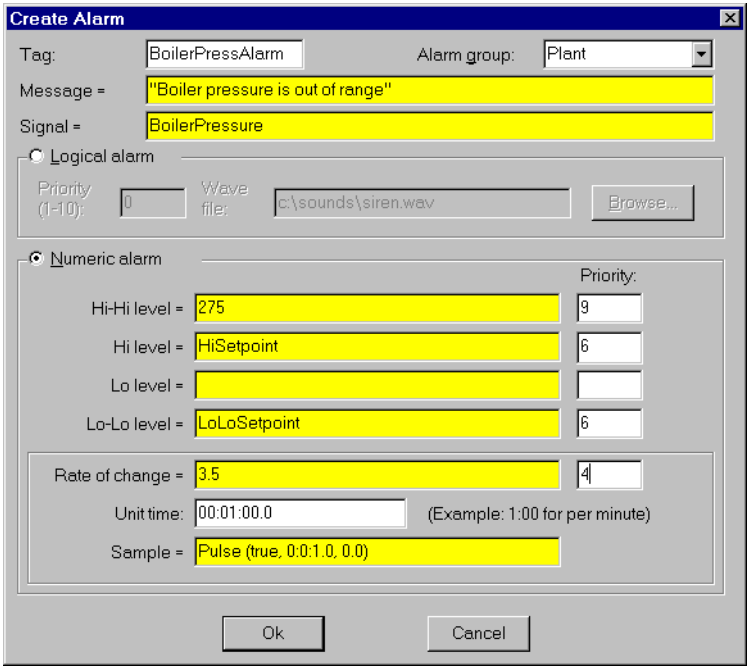


Figure 18-8. Typical Settings for a Numeric Style Alarm

Alarm group specifies the group name associated with the alarm object. All previously defined groups appear in the list box and may be selected with the mouse.

Priority ranges from 1 to 10 where 10 is the most important.

Message is a text expression whose result is displayed in the alarm window when this object generates an alarm. If alarm style is numeric, the relevant alarm trigger prefixes your message (for example, HiHi level: *Alarm message*). See Chapter 9, *Alarms*, for additional information

Alarm Data Members

Table 18-6. Alarm Data Members

Data Members	Type	Read	Write	Description
active	logical	yes	no	Result of logical alarm status. True if alarm is active and false if alarm is inactive.
hi	logical	yes	no	Result of numeric alarm hi data member status.
hihi	logical	yes	no	Result of numeric alarm hihi data member status.
lo	logical	yes	no	Result of numeric alarm lo data member status.
lolo	logical	yes	no	Result of numeric alarm lolo data member status.
rate	logical	yes	no	Result of numeric alarm rate data member status.

Comments ***A common alarm condition is caused by a measured value going out of an acceptable range. For example, a storage tank whose level is too low or too high can generate several alarms:*** `Hihi: Tank level is out of range`***, or*** `Lo: Tank level is out of range`***. If you use a numeric style alarm to trigger the alarm and if the tank level fluctuates or "jitters" around the alarm level settings, a new alarm record is generated in the alarm window each time the tank level fluctuates above or below the level settings. To alleviate this condition, you could use the Neutralzone object to filter out minor tank fluctuations.***

☞ **Note** ***As discussed in Chapters 5 and 9, alarms can also be defined through parameter settings on object data members. Many times, this is the most efficient method of defining and creating alarm conditions. See*** *Database-Generated Alarms* ***in Chapter 9,*** *Alarms****, and*** *Editing Object Databases* in Chapter 4, *Using Lookout****, for more information.***

$Alarm

$Alarm is a global object. It makes available global alarm data such as the number of currently active alarms.

You can use $Alarm data members just like other data members. By inserting an expression you can display the number of active alarms in any particular group. Or, by connecting a pushbutton to an acknowledge data member, operators can acknowledge alarms through pushbutton selection.

Assume, for example, that you want to create a pushbutton that acknowledges all alarms. First create a pushbutton object. Next, use the **Object»Edit Connections…** command to connect the pushbutton (`Pb1`) to the `.ack` data member of $Alarm. Such a connection is shown below:

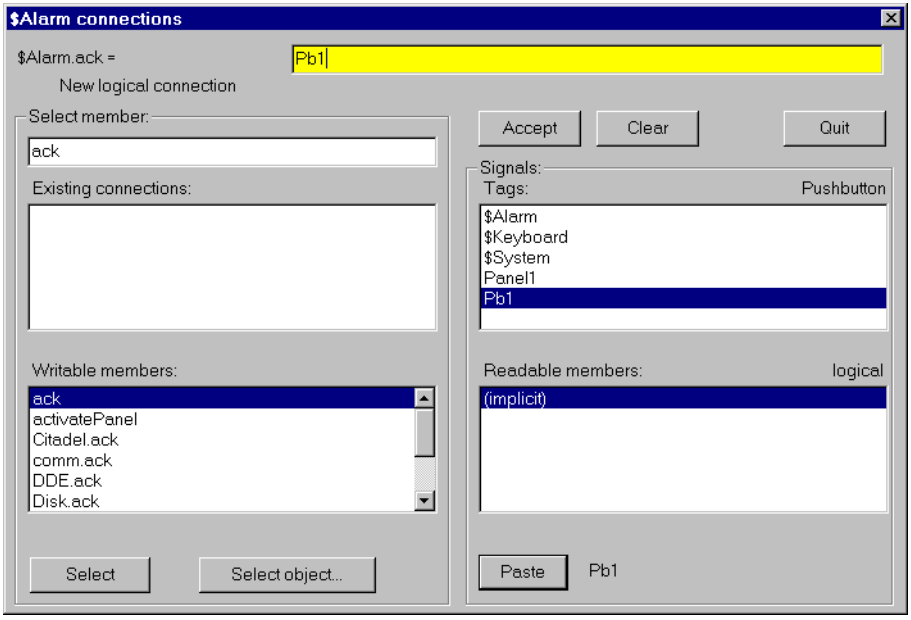


Figure 18-9. Edit Connections Dialog Box for using $Alarm

The expression `(Pb1)` acknowledges all active alarms any time the pushbutton is depressed. You could make similar connections to acknowledge individual alarm groups.

You can define new alarm groups through Alarm objects and by modifying object database parameters. As you create each new alarm group, Lookout

adds new readable and writable data members to the $Alarm object. $Alarm data members are described in the following table.

Alarm Data Members

Table 18-7. $Alarm Data Members

Data Member	Type	Read	Write	Description
ack	logical	no	yes	Upon transition from FALSE to TRUE, acknowledges all alarms.
Active	numeric	yes	no	Total number of currently active alarms (that is, alarm conditions that still exist).
ActivatePanel	logical	no	yes	Calls Alarm Window upon transition from FALSE to TRUE, making it visible on the screen.
Groupname	text	yes	no	Alarm group name associated with the most recent alarm
Groupname.active	numeric	yes	no	Number of currently active alarms within the specified group.
Groupname.unacked	numeric	yes	no	Number of unacknowledged alarms within the specified group.
Groupname.ack	logical	no	yes	Upon transition from FALSE to TRUE, acknowledges all alarms within the specified group.
Message	text	yes	no	Text of the message of the most recent alarm
MinimizePanel	logical	no	yes	Closes Alarm Window upon transition from FALSE to TRUE, changing it to an icon.
Priority	numeric	yes	no	Alarm priority associated with the most recent alarm
Priority01.active – Priority10.active	numeric	yes	no	Number of currently active alarms of the specified priority.
Priority01.unacked – Priority10.unacked	numeric	yes	no	Number of unacknowledged alarms of the specified priority.

Table 18-7. $Alarm Data Members (Continued)

Data Member	Type	Read	Write	Description
Tagname	text	yes	no	Object tag name associated with the most recent alarm
Unacked	numeric	yes	no	Total number of unacknowledged alarms (that is, alarm conditions that have not yet been acknowledged).
Update	logical	yes	no	Pulses high every time there is a new alarm

Comments ***Groupname in the preceding table represents the name of any specified alarm group. The $Alarm object contains an*** `.active`***,*** `.unacked`***, and*** `.ack` ***data member for every alarm group.***

Using $Alarm with Other Objects

Every time a new alarm appears in Lookout, `$Alarm.Message`, `$Alarm.Tagname`, `$Alarm.Groupname`, and `$Alarm.Priority` are updated with the appropriate information for the new alarm. Then `$Alarm.Update` pulses high to alert you that those four data members contain fresh values.

These data members can serve many purposes, but they are specifically designed to work with the Pager object class. In a typical application, you could use `$Alarm.Update` to initiate a page, possibly including the other four data members in the text of the page. With this system you can also filter which alarms you want to page, and which alarms the pager ignores.

Alternator

Alternator is an object class Lookout uses to control a series of devices with various commands. Typically this means rotating the usage of the devices at specified times.

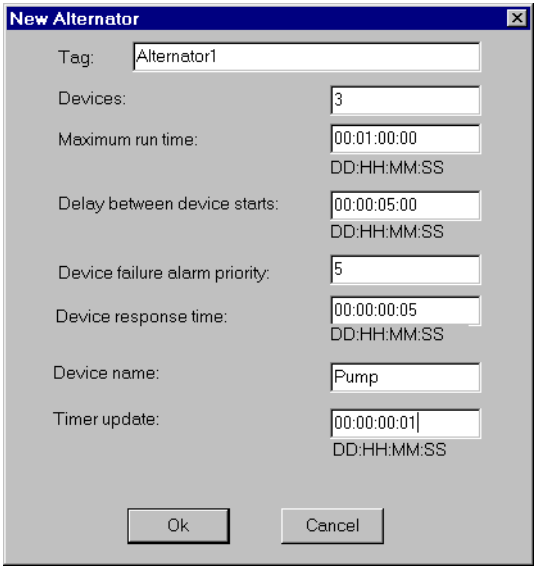


Figure 18-10. Alternator Definition Parameters Dialog Box

Devices determines the maximum number of devices that can be controlled by the alternator.

Maximum run time is the maximum amount of time that any one device is allowed to run continuously. This function can be disabled by entering 0.

Delay between device starts is the minimum amount of time between device starts that are initiated by the alternator object. This function can be disabled by entering 0.

Device failure alarm priority determines the priority level of alarms generated by the alternator. This defaults to 5.

Device response time is the amount of time the alternator waits for a response from a device before activating an alarm. This function can be disabled for individual devices by not connecting anything to the `Response` data member.

Device name is a string describing the type of device that is being controlled by the alternator. This is used to create the alarm messages and is also the name of the group to which the alarms belong.

Timer update specifies the resolution of the timers used by the alternator to decide which devices to turn on and off. This also specifies how often the alternator updates the ElapsedTime and RunTime data members.

Alternator Data Members

Table 18-8. Alternator Data Members

Data Member	Type	Read	Write	Description
ElapsedTime1 - ElapsedTime100	numeric	yes	no	Elapsed time device has been on since last reset
DeviceConnect1 - DeviceConnect100	logical	yes	no	States whether device is in the active sequence considered by the alternator
DeviceOn1 - DeviceOn100	logical	yes	no	Confirmation of the actual device state
DeviceRequest1 - DeviceRequest100	logical	yes	no	Output to whatever drives the device
Advance	logical	no	yes	Turns on the next device in the sequence and turns off the last device in the sequence
Command	numeric	no	yes	How many devices the alternator is to have running
HOAmode1 - HOAmode100	numeric	no	yes	States whether device is in Hand, Off, or Auto mode
Reset1 - Reset100	logical	no	yes	Used to reset the ElapsedTime data member
Response1 - Response100	logical	no	yes	Response bit from device
RunTime1 - RunTime100	numeric	yes	no	Elapsed time device has been on continuously

Connecting the Alternator

The **DeviceRequest** data members should be connected to whatever drivers are controlling your devices. Notice that when you connect the **DeviceRequest** data member to anything, the **DeviceConnect** data member goes high. This connection is the only one required for the alternator to consider your device connected. The **RunTime**, **ElapsedTime**, **DeviceConnect**, and **DeviceOn** data members can be connected to or not depending on whether you want that information.

The **Command** data member is the essential writable data member to connect to. The connection to **Command** can be any numeric expression, but a pot is probably most convenient, especially for testing and demonstration purposes. It only accepts values in the integer range from zero to the number of **Device connections** specified as a parameter. To the **Response** data members should be connected a response bit from each device. If no connection is made to **Response** the alternator does not require any response from the device. The **Advance**, **Reset**, and **HOAmode** data members are not essential either; You need only make connections to them if you want their functionality. **Advance** and **Reset** typically are connected to pushbuttons, and **HOAmode** typically connects to a pot. The **HOAmode** data member only accepts values in the integer range from one to three.

Command and Advance

The **Command** data member is the number of devices that the alternator is to have running at one time. When you send the alternator its first command, it requests a corresponding number of devices to turn on. The **DeviceRequest** data member for each of these devices goes high to signify this. When the devices respond that they are in fact on via the **Response** data member, (if it is connected) the alternator recognizes that the device is on and the **DeviceOn** data member goes high. Conversely, if the **Command** data member specifies that devices are to be turned off, the **DeviceRequest** data members for the corresponding number of devices goes low. The devices should then respond accordingly via the **Response** data member.

When you give the alternator a numeric command, it takes into account the number of devices currently running, and makes its requests accordingly. Because of this, it is important to know how the alternator counts the number of devices running. Devices running as a result of a `Hand` command from the **HOAMode** data member are **NOT** counted. All other running devices (including those with active alarms) are.

When the alternator is asked to turn on a new device it chooses the device that has been off the longest. When asked to turn off a device it chooses the device that has been on the longest.

The **Advance** command turns on one device and turns off another device.

Maximum Run Time and Delay Operation

The **Maximum run time** parameter specifies how long any one device can be running. Each device has its own timer to ensure that this is enforced. When one of these timers informs the alternator that its device has been on for the maximum allowed time, the alternator executes a command identical to the **Advance** command. Notice that there are two cases where the maximum time limit is not enforced strictly. One of them is when you have turned on a device via the HOAmode data member. (Its operation is discussed below.) The other is when it conflicts with the operation of the **Delay between device starts** feature.

Delay between device starts is specified as a parameter, and limits how often the alternator can turn on devices. An example of the operation of the **Delay between device starts** would be beginning with a new alternator that has none of the devices on. You issue a command using the Command data member of 3. If the delay feature is in use, the alternator sends three `ON` requests separated by the amount of time specified by **Delay between device starts**.

Hand - Off - Auto Modes

The **HOAmode** data member gives the user direct control over each device. The integer value of this numeric data member corresponds to an operating mode. (See Table 18-9 below.) When the **HOAmode** data member is in Auto mode (as it is by default if you never connect anything to this data member) it is subject to the commands of the alternator. Its **DeviceConnect** data member will be high to signify that the alternator recognizes it. When you select Hand mode, two things happen: The device is disconnected and is sent an on request. Similarly in Off mode the device is disconnected and requested off. In these two modes, the alternator does not recognize or request anything of these devices. If you select Auto mode again, the device is reconnected.

Table 18-9. HOA Modes

HOA mode	Integer
Hand	1
Off	2
Auto	3

Elapsed Time, Run Time, and Reset

The **ElapsedTime** data member keeps track of how long a particular device has been on since its last reset (or since it was created). Its value is in days, so by viewing it through a numeric expression you may select any time format.

The **RunTime** data member keeps track of how long a particular device has been on continuously. Its value is in days, so by viewing it through a numeric expression you may select any time format.

The **Reset** data member is used to reset **ElapsedTime**.

Alternator Status Messages

Not responding to on request
A device that was sent an ON request via the **DeviceRequest** data member did not respond that it had in fact turned on. The alternator attempts to turn it on again when its turn comes and the alarm remains active until the device responds properly to an alternator command.

Not responding to off request
A device that was sent an OFF request via the **DeviceRequest** data member did not respond that it had in fact turned off. The alternator attempts to turn it off again when its turn comes and the alarm remains active until the device responds properly to an alternator command.

Turned on unexpectedly
A device that was previously off and that the alternator expected to remain off turned on without reason. The alternator attempts to turn it off when its turn comes, and the alarm remains active until the device responds properly to an alternator command.

Turned off unexpectedly
A device that was previously on and that the alternator expected to remain on turned off without reason. The alternator attempts to turn it on when its turn comes, and the alarm remains active until the device responds properly to an alternator command.

Not responding to HOA command
A device did not respond properly to a command from the HOAmode data member. The alternator attempts to request it when its turn comes, and the alarm remains active until the device responds properly to an alternator command.

Animator

The Animator object class provides full graphical animation including horizontal and vertical motion, dynamic resizing and visibility, dynamic symbol sequencing and programmable color changes.

When you first create an animator object, the **Select graphic** dialog box appears.

Figure 18-11. Select Graphic Dialog Box

Select the graphic you want to animate. To animate colors, you must select a Windows Metafile (.WMF). A moving animation must have the series of images you want to animate arranged in a single bitmap so that each cell of the animation can be defined by a grid of rows and columns.

Animations

To create a moving animation, select the Animation tag.

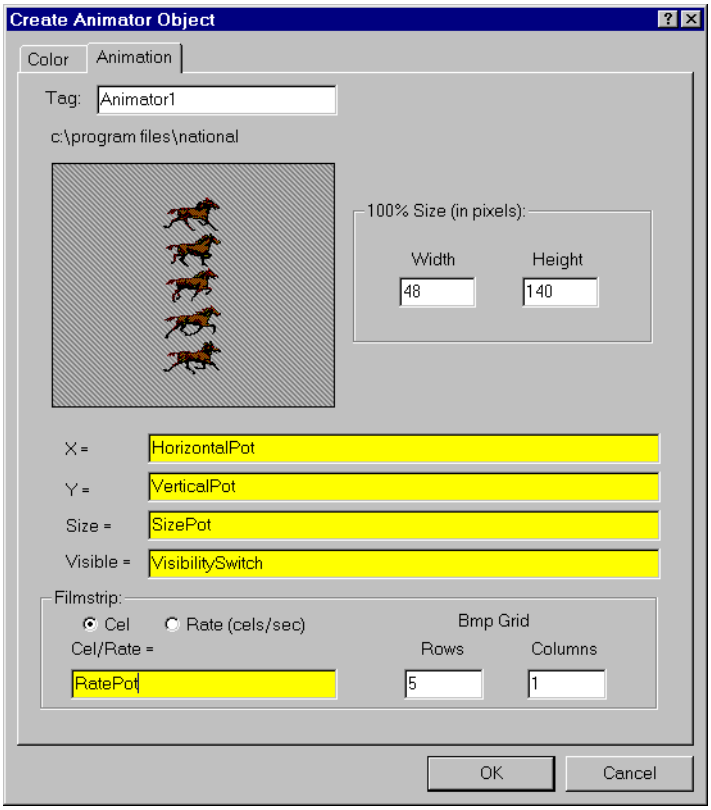


Figure 18-12. Animator Definition Parameters Dialog Box, Animation Tab

The graphic file you select determines what images appear on the screen. You delineate the images in a bitmap graphic by dividing it into a grid of **Rows** and **Columns**. In the example above, the single column is 48 pixels **W**ide and each row is 28 pixels **H**igh. Conceivably, the grid *could* consist of 32,000 cells. 100 cells would be more practical.

Each grid cell is a filmstrip image. Internally, the Animator assigns a cell number to each image. It normally plays the filmstrip by progressing from left to right and top to bottom, from the lowest cell number to the highest cell number.

The **Rate (cells/sec)** selection you use to specify a frequency at which the Animator progresses from one cell image to the next. The rate value can be positive or negative and can range from `0.0000005` (one frame every 23.148 days) to `100` frames per second.

If the rate value is negative, the Animator plays the filmstrip backwards, starting at the last cell.

You can use the **Cell** selection in the definition dialog box to identify a particular cell number to display. For example, you might use this selection to display a specific image when a PLC register is equal to a particular value. This is similar to the multistate object, but can provide many more states, depending on the number of cells in the bitmap.

The **X** and **Y** parameters specify the horizontal and vertical position of the image, respectively. These numeric parameters range from `0` to `100`, and together provide the X-Y position of the image as a percent of the Animator dimension on a control panel.

Size is a numeric parameter that ranges from `0` to `100` percent, where `100` percent is the full size of a single cell as specified by **W** and **H**.

Visible is a logical parameter that causes to the image to appear when it goes true and disappear when it goes false.

Color Animation

To create color changes programmatically, select the Color tab.

With this option, you can set the color map on a .WMF graphic using a series of five logical conditions and a default state. If no condition is true, the default state (or color map) is imposed.

You can choose a custom color map by clicking on the color sample associated with each logical condition. You can then select any color from the palette available on your computer.

Select the Show Original Colors box if you want the original color map of the .WMF file to be imposed for that condition.

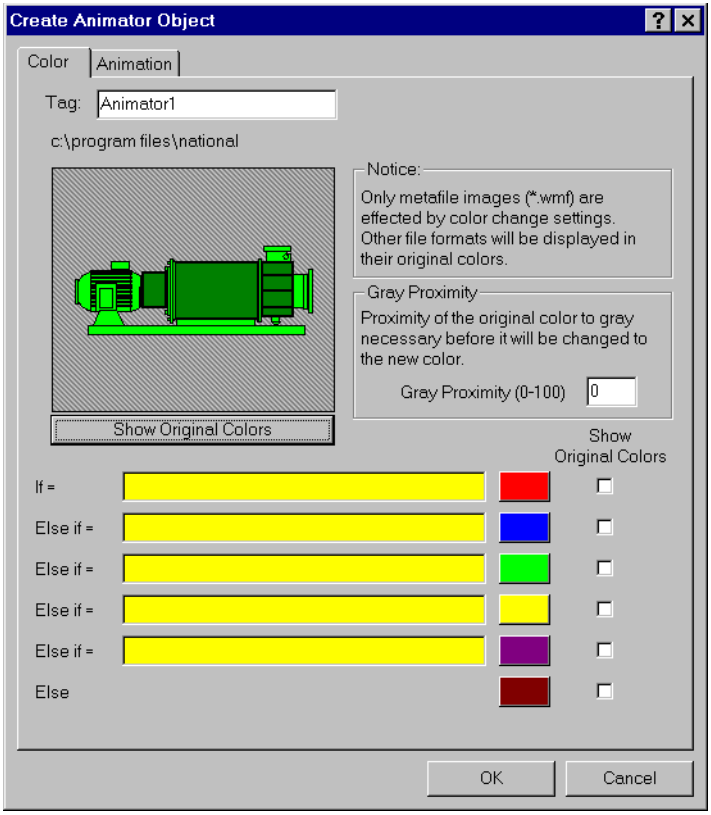


Figure 18-13. Animator Definition Parameters Dialog Box, Color Tab

Gray Proximity determines the color saturation point at which the selected color replaces an original color. When this parameter is set to 0, most colors change. When this parameter is set to 100, most colors will not change. You can test the effects of this setting by putting your cursor over the color button and observing the change in the graphic displayed in the dialog box window.

Notice that the if and else if statements are evaluated in sequence. If several conditional expressions are true at once, Multistate displays the graphic associated with the first true expression.

For instance, if your **If** parameters use less-than comparisons, such as `PumpSpeed<50`, the following **Else if** parameter must have a larger comparison value, such as `PumpSpeed<75`. If you use a smaller comparison value, such as `PumpSpeed<25`, the color change will not take

place. In other words, the comparison values must be used from smallest to largest.

In the same way, if you are using greater-than comparisons, such as `PumpSpeed>50`, you must list your comparison values from largest to smallest, so that the next **Else If** parameter would have to be something like `PumpSpeed>25`.

Put **If** parameters using equality, such as `PumpSpeed=50`, before parameters using inequalities.

A few minutes experimentation should help you understand the interactions of the color choice conditions.

Animator Data Members

Table 18-10. Animator Data Members

Data Member	Type	Read	Write	Description
none	—	—	—	Animator objects do not have data members

Comments ***Consider using an Integral object instead of a Counter/Pulse combination when trying to achieve smooth animation.***

Related Objects ***Multistate, Pipe, Gauge, Spinner, DialGauge***

Applicom

Applicom is a set of object classes Lookout uses to communicate with a series of devices and protocols, including the Applicom on-board database (Local mode), April 1000, Klockner-Moeller, Otic Fischer & Porter, Profibus DP, Profibus FMS, Profibus L2, SAIA-SBus, Siemens S7MPI, Siemens H1, Siemens S5 AS511, and Telemecanique.

Pictured below on the left is the dialog box for creating an Applicom Local object. At right is the dialog box for creating an Applicom Telemecanique object. The Telemecanique object has the same parameters as all the other Applicom objects except for Applicom Local, and makes a good representative sample for the group.

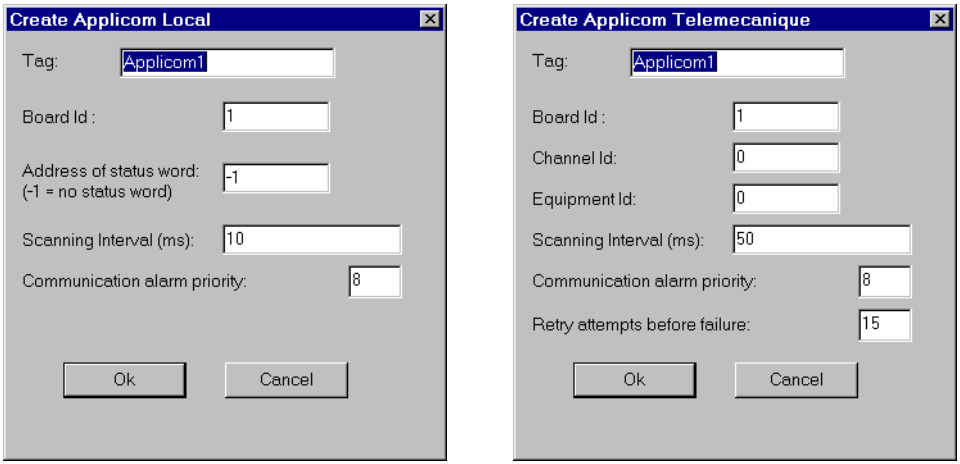


Figure 18-14. Applicom Definition Parameters Dialog Boxes

Board ID identifies the board you are trying to address.

Address of status word is the address of the specific cyclic function of your Applicom board that you want to interact with using this Applicom Local object.

Channel ID identifies the Applicom channel your device is communicating on.

Equipment ID is the Applicom equipment ID for the device you are connecting to.

Scanning interval is the time period between polls. Valid range is 10 - 65535 (expressed in msec).

Communication alarm priority determines the priority level of the alarms generated by the object.

Retry attempts before failure specifies the consecutive number of times Lookout attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the Applicom object generates an alarm and releases the communication port back to the communications service which then moves on to the next device in the polling queue (if any). Refer to Chapter 6, *Serial Communications,* for more detailed information.

Table 18-11. Lookout Applicom Object Classes and the Corresponding Protocols/Devices

Object Class	Device/protocol
Applicom Local	Any device or protocol supported by Applicom cards. See the *Special Instructions on Using the Applicom Local Object Class* section, for more detailed information on using this object.
Applicom JBUS	Devices using Jbus protocol (mostly those made by April).
April 1000	April 1000 PLC using Ethway or Fipway protocols.
Klockner-Moeller	Klockner-Moeller SUCOS PS32 and PS316 PLC using the Sucoma link.
Otic Fischer & Porter	Otic Fischer & Porter controllers of series 2000, 5000, and 2000 supervisor using the DATALINK protocol.
Profibus DP	Profibus DP with any I/O module or PLC that communicates on a Profibus network using the Decentralized Periphery protocol (includes some Siemens S5 PLCs).
Profibus FMS	Profibus FMS protocol with any master or slave FMS device on a Profibus network (includes most Siemens PLCs).
Profibus L2	Profibus L2 protocol with Siemens S5 PLC using Siemens France messaging system. (The PLC must be running a program also distributed with the Applicom hardware.)
SAIA-SBus	SAIA SBus master protocol.
Siemens S7MPI	Siemens MPI protocol with Siemens S7 PLCs.
Siemens H1	Sinec H1 protocol with various Siemens PLCs.

Table 18-11. Lookout Applicom Object Classes and the Corresponding Protocols/Devices (Continued)

Object Class	Device/protocol
Siemens S5 AS511	AS511 (programming port) protocol with Siemens S5 PLCs.
Telemecanique	Uni-Telway, Ethway and Fipway protocols with all Telemecanique PLCs.

Applicom Data Members

Table 18-12. Applicom Local Data Members

Data Member	Type	Read	Write	Description
B0-B31999	logical	yes	yes	output bits
W0-W31999	numeric	yes	yes	output words
D0-D31998	numeric	yes	yes	internal double words
F0-F31998	numeric	yes	yes	internal floating words
O0-O13999	numeric	yes	yes	internal bytes
WS0-WS31999	numeric	yes	yes	signed output words
DS0-DS31998	numeric	yes	yes	signed internal double words
OS0-OS13999	numeric	yes	yes	signed internal bytes
Update	logical	yes	no	goes FALSE when a poll starts and TRUE when a poll completes
CommFail	logical	yes	no	object-generated signal that is ON if Lookout cannot communicate with the device(s)

Table 18-13. Applicom JBUS Data Members

Data Member	Type	Read	Write	Description
B0-B4294967295	logical	yes	yes	internal bits
BI0-BI4294967295	logical	yes	no	input bits
BO0-BO4294967295	logical	yes	yes	output bits
O0-O4294967295	numeric	yes	yes	internal bytes

Table 18-13. Applicom JBUS Data Members (Continued)

Data Member	Type	Read	Write	Description
OI0-OI4294967295	numeric	yes	no	input bytes
OO0-OO4294967295	numeric	yes	yes	output bytes
W0-W4294967295	numeric	yes	yes	internal words
WI0-WI4294967295	numeric	yes	no	input words
WO0-WO4294967295	numeric	yes	yes	output words
OS0-OS4294967295	numeric	yes	yes	signed internal bytes
OIS0-OIS4294967295	numeric	yes	no	signed input bytes
OOS0-OOS4294967295	numeric	yes	yes	signed output bytes
WS0-WS4294967295	numeric	yes	yes	signed internal words
WIS0-WIS4294967295	numeric	yes	no	signed input words
WOS0-WOS4294967295	numeric	yes	yes	signed output words
O0.0-O4294967295.7	logical	yes	yes	bit in internal bytes
O0.0-O4294967295.7	logical	yes	yes	bit in internal bytes
W0.0-W4294967295.15	logical	yes	yes	bits in internal words
D0-D4294967295	numeric	yes	yes	internal double words
F0-F4294967295	numeric	yes	yes	internal floating words
Update	logical	yes	no	goes FALSE when a poll starts and TRUE when a poll completes
CommFail	logical	yes	no	object-generated signal that is ON if Lookout cannot communicate with the device(s)

Table 18-14. Applicom April 1000 Data Members

Data Member	Type	Read	Write	Description
MX0-MX16383	logical	yes	yes	not safeguarded internal bits
RX0-RX8191	logical	yes	yes	safeguarded internal bits
IX00000-IX96931	logical	yes	no	input bits

Table 18-14. Applicom April 1000 Data Members (Continued)

Data Member	Type	Read	Write	Description
QX00000-QX96931	logical	yes	yes	output bits
MW0-MW409599	numeric	yes	yes	data words
MWS0-MWS409599	numeric	yes	yes	signed data words
MD0-MD409598	numeric	yes	yes	double words in the data words
MDS0-MDS409598	numeric	yes	yes	signed double words in the data words
FD0-FD409598	numeric	yes	yes	floating words in the data words
Update	logical	yes	no	goes FALSE when a poll starts and TRUE when a poll completes
CommFail	logical	yes	no	object-generated signal that is ON if Lookout cannot communicate with the device(s)

Table 18-15. Applicom Klockner-Moeller Data Members

Data Member	Type	Read	Write	Description
B0-B4294967295	logical	yes	yes	internal bits
BI0-BI4294967295	logical	yes	no	input bits
BO0-BO4294967295	logical	yes	yes	output bits
O0-O4294967295	numeric	yes	yes	internal bytes
OI0-OI4294967295	numeric	yes	no	input bytes
OO0-OO4294967295	numeric	yes	yes	output bytes
W0-W4294967295	numeric	yes	yes	internal words
WI0-WI4294967295	numeric	yes	no	input words
WO0-WO4294967295	numeric	yes	yes	output words
OS0-OS4294967295	numeric	yes	yes	signed internal bytes
OIS0-OIS4294967295	numeric	yes	no	signed input bytes
OOS0-OOS4294967295	numeric	yes	yes	signed output bytes
WS0-WS4294967295	numeric	yes	yes	signed internal words

Table 18-15. Applicom Klockner-Moeller Data Members (Continued)

Data Member	Type	Read	Write	Description
WIS0-WIS4294967295	numeric	yes	no	signed input words
WOS0-WOS4294967295	numeric	yes	yes	signed output words
O0.0-O4294967295.7	logical	yes	yes	bit in internal bytes
W0.0-W4294967295.15	logical	yes	yes	bits in internal words
D0-D4294967295	numeric	yes	yes	internal double words
DS0-DS4294967295	numeric	yes	yes	signed internal double words
F0-F4294967295	numeric	yes	yes	internal floating words
Update	logical	yes	no	goes FALSE when a poll starts and TRUE when a poll completes
CommFail	logical	yes	no	object-generated signal that is ON if Lookout cannot communicate with the device(s)

Table 18-16. Applicom Otic Fischer & Porter Data Members

Data Member	Type	Read	Write	Description
L0-L65535	logical	yes	yes	bits
B0-B65535	numeric	yes	yes	bytes
BS0-BS65535	numeric	yes	yes	Signed Bytes
B0.0-B65535.7	logical	yes	yes	bits in bytes
C0-C65535	numeric	yes	yes	floating word on three bytes
H0-H65535	numeric	yes	yes	floating word on five bytes
Update	logical	yes	no	goes FALSE when a poll starts and TRUE when a poll completes
CommFail	logical	yes	no	object-generated signal that is ON if Lookout cannot communicate with the device(s)

Table 18-17. Applicom Profibus DP Data Members

Data Member	Type	Read	Write	Description
B0-B4294967295	logical	yes	yes	internal bits
BI0-BI4294967295	logical	yes	no	input bits
BO0-BO4294967295	logical	yes	yes	output bits
O0-O4294967295	numeric	yes	yes	internal bytes
OI0-OI4294967295	numeric	yes	no	input bytes
OO0-OO4294967295	numeric	yes	yes	output bytes
W0-W4294967295	numeric	yes	yes	internal words
WI0-WI4294967295	numeric	yes	no	input words
WO0-WO4294967295	numeric	yes	yes	output words
OS0-OS4294967295	numeric	yes	yes	signed internal bytes
OIS0-OIS4294967295	numeric	yes	no	signed input bytes
OOS0-OOS4294967295	numeric	yes	yes	signed output bytes
WS0-WS4294967295	numeric	yes	yes	signed internal words
WIS0-WIS4294967295	numeric	yes	no	signed input words
WOS0-WOS4294967295	numeric	yes	yes	signed output words
O0.0-O4294967295.7	logical	yes	yes	bit in internal bytes
W0.0-W4294967295.15	logical	yes	yes	bits in internal words
D0-D4294967295	numeric	yes	yes	internal double words
DS0-DS4294967295	numeric	yes	yes	signed internal double words
F0-F4294967295	numeric	yes	yes	internal floating words
Update	logical	yes	no	goes FALSE when a poll starts and TRUE when a poll completes
CommFail	logical	yes	no	object-generated signal that is ON if Lookout cannot communicate with the device(s)

Table 18-18. Applicom Profibus FMS Data Members

Data Member	Type	Read	Write	Description
BID0-BID65535	logical	yes	yes	bits (simple variables)
BID0SUB0-BID65535SUB240	logical	yes	yes	bits
OID0-OID65535	numeric	yes	yes	bytes (simple variables)
OID0SUB0-OID65535SUB240	numeric	yes	yes	bytes
WID0-WID65535	numeric	yes	yes	words (simple variables)
WID0SUB0-WID65535SUB120	numeric	yes	yes	words
DID0-DID65535	numeric	yes	yes	doubles words (simple variables)
DID0SUB0-DID65535SUB60	numeric	yes	yes	doubles words
FID0-FID65535	numeric	yes	yes	floating words (simple variables)
FID0SUB0-FID65535SUB60	numeric	yes	yes	floating words
OIDS0-OIDS65535	numeric	yes	yes	signed bytes (simple variables)
OID0SUBS0-OID65535SUBS240	numeric	yes	yes	signed bytes
WIDS0-WIDS65535	numeric	yes	yes	signed words (simple variables)
WID0SUBS0-WID65535SUBS120	numeric	yes	yes	signed words
Update	logical	yes	no	goes FALSE when a poll starts and TRUE when a poll completes
CommFail	logical	yes	no	object-generated signal that is ON if Lookout cannot communicate with the device(s)

Table 18-19. Applicom Profibus L2 Data Members

Data Member	Type	Read	Write	Description
F0.0-F255.7	logical	yes	yes	bits in flag bytes
S0.0-S4095.7	logical	yes	yes	bits in Sflags (internal bytes)
I0.0-I127.7	logical	yes	no	input bits
Q0.0-Q127.7	logical	yes	yes	output bits
DB1D0.0-DB255D255.15	logical	yes	yes	bits of DB word
DX0D0.0-DX255D255.15	logical	yes	yes	bits of DX word
FY0-FY255	numeric	yes	yes	flag bytes
SY0-SY4095	numeric	yes	yes	Sflags (internal bytes)
IB0-IB127	numeric	yes	no	input bytes
QY0-QY127	numeric	yes	yes	output bytes
FW0-FW254	numeric	yes	yes	words in flag bytes
SW0-SW4094	numeric	yes	yes	words in Sflags (internal bytes)
IW0-IW126	numeric	yes	no	input words
QW0-QW126	numeric	yes	yes	output words
DB1DW0-DB255DW255	numeric	yes	yes	Update pulse: words in DB
DX0DW0-DX255DW255	numeric	yes	yes	Update pulse: words in DX
FD0-FD255	numeric	yes	yes	double words in flag bytes
SD0-SD4095	numeric	yes	yes	floating words in Sflags
DB1DD0-DB255DD254	numeric	yes	yes	double words in DB
DX0DD0-DX255DD254	numeric	yes	yes	double words in DX
FD0KG-FD252KG	numeric	yes	yes	floating words in flag bytes
SD0KG-SD4092KG	numeric	yes	yes	floating words in Sflags
DB1DD0KG-DB255DD254KG	numeric	yes	yes	floating words in DB
DX0DD0KG-DX255DD254KG	numeric	yes	yes	floating words in DX

Table 18-19. Applicom Profibus L2 Data Members (Continued)

Data Member	Type	Read	Write	Description
FYS0-FYS255	numeric	yes	yes	signed flag bytes
SYS0-SYS4095	numeric	yes	yes	signed Sflags (internal bytes)
IBS0-IBS127	numeric	yes	no	signed input bytes
QYS0-QYS127	numeric	yes	yes	signed output bytes
FWS0-FWS254	numeric	yes	yes	signed words in flag bytes
SWS0-SWS4094	numeric	yes	yes	signed words in Sflags (internal bytes)
IWS0-IWS126	numeric	yes	no	signed input words
QWS0-QWS126	numeric	yes	yes	signed output words
DB1DWS0-DB255DWS255	numeric	yes	yes	signed words in DB
DX0DWS0-DX255DWS255	numeric	yes	yes	signed words in DX
FDS0-FDS255	numeric	yes	yes	signed double words in flag bytes
SDS0-SDS4095	numeric	yes	yes	signed double words in Sflags
DB1DDS0-DB255DDS254	numeric	yes	yes	signed double words in DB
DX0DDS0-DX255DDS254	numeric	yes	yes	signed double words in DX
Update	logical	yes	no	goes FALSE when a poll starts and TRUE when a poll completes
CommFail	logical	yes	no	object-generated signal that is ON if Lookout cannot communicate with the device(s)

Table 18-20. Applicom SAIA SBus Data Members

Data Member	Type	Read	Write	Description
F0-F8091	logical	yes	yes	flags
I0-I8091	logical	yes	no	input
O0-O8091	logical	yes	yes	output
R0-R4095	numeric	yes	yes	binary registers
RS0-RS4095	numeric	yes	yes	signed binary registers
R0.0-R4095.31	logical	yes	yes	bits in binary registers
Q0-Q4095	numeric	yes	yes	floating registers
DB0R0-DB7999R7999	numeric	yes	yes	binary data block
DB0R0.0-DB7999R7999.31	logical	yes	yes	bits in binary data block
DB0R0.0-DB7999R7999.31	logical	yes	yes	bits in binary data block
DB0Q0-DB7999Q7999	numeric	yes	yes	floating data block
Update	logical	yes	no	goes FALSE when a poll starts and TRUE when a poll completes
CommFail	logical	yes	no	object-generated signal that is ON if Lookout cannot communicate with the device(s)

Table 18-21. Applicom Siemens S5 AS511 Data Members

Data Member	Type	Read	Write	Description
F0.0-F255.7	logical	yes	yes	bits in flag bytes
S0.0-S4095.7	logical	yes	yes	bits in Sflags (internal bytes)
I0.0-I127.7	logical	yes	no	input bits
Q0.0-Q127.7	logical	yes	yes	output bits
DB1D0.0-DB255D255.15	logical	yes	yes	bits of DB word
DX0D0.0-DX255D255.15	logical	yes	yes	bits of DX word
FY0-FY255	numeric	yes	yes	flag bytes
SY0-SY4095	numeric	yes	yes	Sflags (internal bytes)
IB0-IB127	numeric	yes	no	input bytes
QY0-QY127	numeric	yes	yes	output bytes
FW0-FW254	numeric	yes	yes	words in flag bytes
SW0-SW4094	numeric	yes	yes	words in Sflags (internal bytes)
IW0-IW126	numeric	yes	no	input words
QW0-QW126	numeric	yes	yes	output words
DB1DW0-DB255DW255	numeric	yes	yes	Update pulse: words in DB
DX0DW0-DX255DW255	numeric	yes	yes	Update pulse: words in DX
FD0-FD255	numeric	yes	yes	double words in flag bytes
SD0-SD4095	numeric	yes	yes	floating words in Sflags
DB1DD0-DB255DD254	numeric	yes	yes	double words in DB
DX0DD0-DX255DD254	numeric	yes	yes	double words in DX
FD0KG-FD252KG	numeric	yes	yes	floating words in flag bytes
SD0KG-SD4092KG	numeric	yes	yes	floating words in Sflags
DB1DD0KG-DB255DD254KG	numeric	yes	yes	floating words in DB
DX0DD0KG-DX255DD254KG	numeric	yes	yes	floating words in DX
FYS0-FYS255	numeric	yes	yes	signed flag bytes
SYS0-SYS4095	numeric	yes	yes	signed Sflags (internal bytes)

Table 18-21. Applicom Siemens S5 AS511 Data Members (Continued)

Data Member	Type	Read	Write	Description
IBS0-IBS127	numeric	yes	no	signed input bytes
QYS0-QYS127	numeric	yes	yes	signed output bytes
FWS0-FWS254	numeric	yes	yes	signed words in flag bytes
SWS0-SWS4094	numeric	yes	yes	signed words in Sflags (internal bytes)
IWS0-IWS126	numeric	yes	no	signed input words
QWS0-QWS126	numeric	yes	yes	signed output words
DB1DWS0-DB255DWS255	numeric	yes	yes	signed words in DB
DX0DWS0-DX255DWS255	numeric	yes	yes	signed words in DX
FDS0-FDS255	numeric	yes	yes	signed double words in flag bytes
SDS0-SDS4095	numeric	yes	yes	signed double words in Sflags
DB1DDS0-DB255DDS254	numeric	yes	yes	signed double words in DB
DX0DDS0-DX255DDS254	numeric	yes	yes	signed double words in DX
Update	logical	yes	no	goes FALSE when a poll starts and TRUE when a poll completes
CommFail	logical	yes	no	object-generated signal that is ON if Lookout cannot communicate with the device(s)

Table 18-22. Applicom Siemens S7 MPI Data Members

Data Member	Type	Read	Write	Description
M0.0-M65535.7	logical	yes	yes	bits in flag bytes
DB1.DBX0.0-DB8191. DB65535.7	logical	yes	yes	bits in DB byte
E0.0-E65535.7	logical	yes	no	input bits
A0.0-A65535.7	logical	yes	yes	output bits

Table 18-22. Applicom Siemens S7 MPI Data Members (Continued)

Data Member	Type	Read	Write	Description
MB0-MB65535	numeric	yes	yes	flag bytes
DB0DBB0-DB32767. DBB65535	numeric	yes	yes	bytes in DB
EB0-EB65535	numeric	yes	no	input bytes
AB0-AB65535	numeric	yes	yes	output bytes
MW0-MW65534	numeric	yes	yes	words in flag bytes
DB0.DBW0-DB32767. DBW65534	numeric	yes	yes	words in DB
EW0-EW65534	numeric	yes	no	input words
AW0-AW65534	numeric	yes	yes	output words
MD0-MD65532	numeric	yes	yes	double words in flag bytes
DB0.DBD0-DB32767. DBD65532	numeric	yes	yes	double words in DB
MBS0-MBS65535	numeric	yes	yes	signed flag bytes
DB0DBBS0-DB32767. DBBS65535	numeric	yes	yes	signed bytes in DB
EBS0-EBS65535	numeric	yes	no	signed input bytes
ABS0-ABS65535	numeric	yes	yes	signed output bytes
MWS0-MWS65534	numeric	yes	yes	signed words in flag bytes
DB0.DBWS0-DB32767. DBWS65534	numeric	yes	yes	signed words in DB
EWS0-EWS65534	numeric	yes	no	signed input words
AWS0-AWS65534	numeric	yes	yes	signed output words
MDS0-MDS65532	numeric	yes	yes	signed double words in flag bytes
DB0.DBDS0-DB32767. DBDS65532	numeric	yes	yes	signed double words in DB
MD0F-MD65532F	numeric	yes	yes	floating words in flag bytes

Table 18-22. Applicom Siemens S7 MPI Data Members (Continued)

Data Member	Type	Read	Write	Description
DB0.DBD0F-DB32767. DBD65532F	numeric	yes	yes	floating words in DB
Update	logical	yes	no	goes FALSE when a poll starts and TRUE when a poll completes
CommFail	logical	yes	no	object-generated signal that is ON if Lookout cannot communicate with the device(s)

Table 18-23. Applicom Siemens S7 PPI Data Members

Data Member	Type	Read	Write	Description
M0.0-M65535.7	logical	yes	yes	bits in flag bytes
DB1.DBX0.0-DB8191. DB65535.7	logical	yes	yes	bits in DB byte
E0.0-E65535.7	logical	yes	no	input bits
A0.0-A65535.7	logical	yes	yes	output bits
MB0-MB65535	numeric	yes	yes	flag bytes
DB0DBB0-DB32767. DBB65535	numeric	yes	yes	bytes in DB
MBS0-MBS65535	numeric	yes	yes	signed flag bytes
DB0DBBS0-DB32767. DBBS65535	numeric	yes	yes	signed bytes in DB
EBS0-EBS65535	numeric	yes	no	signed input bytes
ABS0-ABS65535	numeric	yes	yes	signed output bytes
MWS0-MWS65534	numeric	yes	yes	signed words in flag bytes
DB0.DBWS0-DB32767. DBWS65534	numeric	yes	yes	signed words in DB
EWS0-EWS65534	numeric	yes	no	signed input words
AWS0-AWS65534	numeric	yes	yes	signed output words
MDS0-MDS65532	numeric	yes	yes	signed double words in flag bytes

Table 18-23. Applicom Siemens S7 PPI Data Members (Continued)

Data Member	Type	Read	Write	Description
DB0.DBDS0-DB32767. DBDS65532	numeric	yes	yes	signed double words in DB
EB0-EB65535	numeric	yes	no	input bytes
AB0-AB65535	numeric	yes	yes	output bytes
MW0-MW65534	numeric	yes	yes	words in flag bytes
DB0.DBW0-DB32767. DBW65534	numeric	yes	yes	words in DB
EW0-EW65534	numeric	yes	no	input words
AW0-AW65534	numeric	yes	yes	output words
MD0-MD65532	numeric	yes	yes	double words in flag bytes
DB0.DBD0-DB32767. DBD65532	numeric	yes	yes	double words in DB
MD0F-MD65532F	numeric	yes	yes	floating words in flag bytes
DB0.DBD0F-DB32767. DBD65532F	numeric	yes	yes	floating words in DB
Update	logical	yes	no	goes FALSE when a poll starts and TRUE when a poll completes
CommFail	logical	yes	no	object-generated signal that is ON if Lookout cannot communicate with the device(s)

Table 18-24. Applicom Siemens H1 Data Members

Data Member	Type	Read	Write	Description
C1-C2097152	logical	yes	yes	Bits
X1-X2097152	logical	yes	no	Input bits
Y1-Y2097152	logical	yes	yes	Output bits
V1-V2097152	numeric	yes	yes	Internal words
VS1-VS2097152	numeric	yes	yes	Signed internal words

Table 18-24. Applicom Siemens H1 Data Members (Continued)

Data Member	Type	Read	Write	Description
V1.0-V2097152.15	logical	yes	yes	Bits in internal words
VD1-VD2097151	numeric	yes	yes	Internal double words
VDS1-VDS2097151	numeric	yes	yes	Signed internal double words
K1-K2097152	numeric	yes	yes	Constant words
KS1-KS2097152	numeric	yes	yes	Signed constant words
K1.0-K2097152.15	logical	yes	yes	Bits in constant words
VF1-VF2097151	numeric	yes	yes	Floating words
KF1-KF2097151	numeric	yes	yes	Floating constant words
WX1-WX2097152	numeric	yes	yes	Input words
WXS1-WXS2097152	numeric	yes	yes	Signed input words
WY1-WY2097152	numeric	yes	yes	Output words
WSY1-WYS2097152	numeric	yes	yes	Signed output words
TCP1-TCP65536	numeric	yes	yes	Timer/counter preset
TCC1-TCC65536	numeric	yes	yes	Timer/counter current
DSP1-DSP65536	numeric	yes	yes	Drum step preset
DSC1-DSC65536	numeric	yes	yes	Drum step current
DCP1-DCP65536	numeric	yes	yes	Drum count preset
STW1-STW65536	numeric	yes	yes	System status words
Update	logical	yes	no	goes FALSE when a poll starts and TRUE when a poll completes
CommFail	logical	yes	no	object-generated signal that is ON if Lookout cannot communicate with the device(s)

Table 18-25. Applicom Telemecanique Data Members

Data Member	Type	Read	Write	Description
B0-B4095	logical	yes	yes	internal bits
I00.0-IF7.F	logical	yes	no	input bits
O00.0-OF7.F	logical	yes	yes	output bits
W0-W32767	numeric	yes	yes	internal words
IW00.0-IWF7.7	numeric	yes	no	words in input registers
OW00.0-OWF7.7 Special	numeric	yes	yes	words in output registers
DW0-DW32766	numeric	yes	yes	internal double words
WS0-WS32767	numeric	yes	yes	signed internal words
IWS00.0-IWSF7.7	numeric	yes	no	signed words in input registers
OWS00.0-OWSF7.7 Special	numeric	yes	yes	signed words in output registers
DWS0-DWS32766	numeric	yes	yes	signed internal double words
FW0-FW32766	numeric	yes	yes	internal floating words
Update	logical	yes	no	goes FALSE when a poll starts and TRUE when a poll completes
CommFail	logical	yes	no	object-generated signal that is ON if Lookout cannot communicate with the device(s)

General Information on Using the Applicom drivers for Lookout

The Applicom drivers for Lookout are all contained in `APPLICOM.CBX` (16- or 32-bit) which is designed to work with the Applicom server software distributed with every Applicom card. These drivers assume that you have version 2.9 or later of the Applicom software installed on the same computer under the same operating system as your copy of Lookout. The `APPLICOM.CBX` file presently supports 12 different object classes corresponding to 12 different protocols or object types.

Before you start using Lookout with Applicom, you must follow certain configuration and testing procedures to ensure that Lookout can communicate with your devices. The steps include configuration, loading, testing, and if you choose, cyclic function configuration.

First, however, take special notice of the two different modes for using the Lookout Applicom driver–Image and Local. The distinction between these two modes is crucial to using the Lookout Applicom object classes.

Applicom Local and Image Modes

You can configure the Applicom server software to constantly poll a remote device and store data in a database onboard your Applicom card. You use *Local* mode to access that data.

In Lookout, you address the data using a generic addressing scheme (as a bit number, or word number, and so on.). The Applicom card takes care of polling the device and writing out new values to registers on the device. If there are any problems in communicating with the device, an error code may be generated and stored in a specified database location. The Lookout driver can then use the data at that location to generate alarms.

In *Image* mode you address data on the remote device according to the addressing scheme commonly used with the particular class of devices. In this mode, Lookout asks for data directly from a remote device and is notified of communication errors when they arise. The Lookout Applicom driver then retries the particular data transfer repeatedly, generating alarms when the data transfer continues to fail.

You use the Applicom Local object class for local mode functionality, and the other device- or protocol-specific object classes for image mode functionality. You can use Applicom Local concurrently with any of the other Applicom classes.

The Image mode classes are the typical way to use the Applicom driver for most applications. However, the Local mode provides a convenient way to achieve two goals: providing access to data from multiple applications (because any application may use the Applicom functions to access the local mode database); and allowing inputs from one device connected to a card to be written to outputs on another device connected to the same card without the interference of the Windows operating system or the client application (for example, Lookout).

Configuration of the Applicom Server

After you install the Applicom server software, you must first configure it using the `PCConf` utility provided as part of the Applicom software. Refer to the *Software Installation* section of your Applicom documentation for details on using Applicom software. You use the `PCConf` utility to set the basic types of devices connected to the Applicom card and the protocols used to connect to them. You also set all the communication parameters with this program.

While configuring your system with `PCConf`, make note of the **Channel** number and **Equipment** number for the device(s) you are using Lookout to communicate with. You will need these values as parameters when you create Lookout objects in Image mode.

Loading of the Applicom Server

After you configure the Applicom Server, you must download the appropriate firmware module to the card and then configure it. You do this by running the Applicom program `PCInit`. The configuration software notifies you at this time if there is any physical problem with the installation of the card.

You must run `PCInit` every time you restart the computer if you plan to communicate with the Applicom card and connected devices during that session. Because of this, you may consider putting a shortcut to this program in the Startup program group.

At this point, the Applicom server should be ready to communicate with Lookout and with the remote devices connected to the card. Before starting Lookout and attempting to communicate, you should first test the connections using some of the Applicom programs.

Testing the Applicom Server

The `ReadWait` program in the Applicom server group is a good example of a program that can be used to test the connectivity of the Applicom card with a remote device. You can also use `WritWait`, `Essai`, and `Essaigb`. Try performing read or write operations using these programs to test whether you can perform basic data transfer operations between the card and the devices. If these tests fail, Lookout will not be able to communicate with the devices either.

If your tests succeed and Lookout fails to communicate with the Applicom devices, it is likely that the problem is in the Lookout object configuration.

Because the Applicom programs require you to use the channel and equipment numbers from the PCConf program, testing helps you verify that you have the correct numbers.

Creating the Cyclic Functions

There is only one more step before you start using the Lookout drivers, and it is required only if you are planning to use the Applicom Local object class. In that case, you must first configure the cyclic functions that exchange data between the onboard database of the Applicom card and the remote devices.

You do this by using either the Applicom CreateCyc or PCCyc utility programs. These programs require that you enter a channel number, an equipment number, and whether the required operation is a read or a write. You also enter the address of the data on the remote device and the requested address of the data in the database. Once you have created and saved the cyclic functions, you can test them using the GetDB and SetDB programs in the Applicom server.

After you verify that the card can communicate with the remote device, you can start using Lookout. Create a Lookout object of the appropriate class, enter the required parameters, and start communicating with the remote device.

Special Instructions on Using the Applicom Local Object Class

The Applicom Local object class behaves very differently from the Image mode object classes. To use it, you must have configured cyclic functions on your Applicom card using the Applicom PCCyc or CreateCyc programs.

After you configure these functions, you can create one or more Applicom Local objects in Lookout to read from or write to the Applicom database directly. The syntax is a generic syntax which does not correspond with the syntax you use with the PLC. The address you specify in Lookout must be the address of the data on the Applicom board, not the address on the remote device.

You can also specify a status word in the PCCyc program for each cyclic function you create. You can then specify a single status address in the object creation/modification dialog box for Applicom Local to get Lookout alarms any time the corresponding cyclic function is unable to exchange data successfully.

The Applicom Local object class is compatible with the other object classes so that you can create multiple Applicom object classes of any type and use them all at the same time.

Applicom Status Messages

The requested address is incorrect
You have requested data for an invalid address. Check the address and try again. For further details check the Applicom documentation for an explanation of error code 2.

Incorrect data type
The type of the data on the device is not compatible with the requested type. This may occur for different reasons in different protocols. For further details check the Applicom documentation for an explanation of error code 3.

Irretrievable data
The data you are trying to access is irretrievable. Please check that the device is connected and has all the appropriate models. For further details check the Applicom documentation for an explanation of error code 4.

Response timed out
The device did not respond to a message within the time-out period. Check that the device is connected. For further details, check the Applicom documentation for an explanation of error code 33.

Check word or parity fault
Check the channel and target equipment configuration and the wiring to the equipment. For further details, check the Applicom documentation for an explanation of error code 34.

Data not available in cyclic read
There was an error in reading from the card. Contact National Instruments technical support if this error occurs.

Equipment not configured
Configure the equipment with PCConf. If you have already done so, check that you are using the correct (logical) equipment ID, as configured in PCConf.

Deferred read or write request when deferred request register is full
There was an error in writing to the card. Contact National Instruments technical support if this error occurs.

Deferred request transfer attempt with transdif when request register is empty
There was an error in writing to the card. Contact National Instruments technical support if this error occurs.

Communication software not initialized
The Applicom board was not initialized using the PCInit software. Close Lookout, run PCInit and try running Lookout again.

Board number not configured
The board number you are attempting to use is not one that has been configured. Please check the configuration using PCConf.

No Applicom interface
There is no Applicom interface present. Check that PCInit executed correctly.

Timeout elapsed. Message lost
A communication problem occurred that may be related to the configuration of the remote equipment or the bus that the equipment is on. For further details, check the Applicom documentation for an explanation of error code 55.

Negative acknowledgment (NAK) from equipment
A message was received by the destination equipment but not processed due to lack of resources. For further details, check the Applicom documentation for an explanation of error code 56.

Communication refused by the equipment.
The equipment is not responding to communication requests. This may be caused by a misconfigured device or by a bad connection with the Applicom card. Check the device configuration, check the Applicom configuration and use the Applicom utilities to check that there is communication between the two.

Bad frame received
The remote device is not behaving correctly. Check the configuration of the device.

Errorcode ID: refer to Applicom documentation
Your application has returned an error specific to your Applicom device. Consult your Applicom documentation for further information.

Aquatrol

Aquatrol is a protocol driver class Lookout uses to communicate with Aquatrol W1500 controllers.

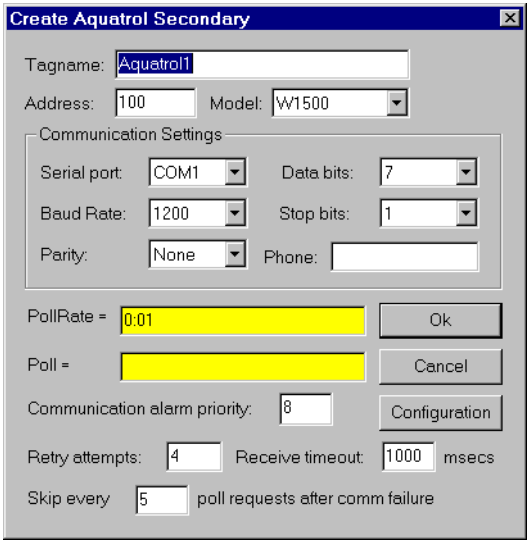


Figure 18-15. Aquatrol Definition Parameters Dialog Box

Address is the address of the RTU as configured on the device. The address can be in the range of 100 to 999 inclusive.

Model is W1500 only at this time.

Serial port specifies which port the object uses for communication to the external device. This does not specify the communication type. Communication type is determined by the **Options»Serial Ports…** command.

Data rate, **Parity**, **Data bits**, and **Stop bits** reference the settings on the hardware device.

Phone specifies the number to be dialed if the serial port setting is configured for dial-up. This number only applies to the individual protocol object.

PollRate is a numeric expression that determines how often to poll the device. Lookout converts the numeric value of **PollRate** into a time signal that represents days and fractions of a day. Aquatrol then polls the device

at the specified time interval. Normally, this is a time constant such as 0:01 (one second). See the *Numeric Data Members* section in Chapter 2, *Introduction*, for information on entering time constants.

Poll is a logical expression. When this expression changes from false to true, Lookout polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Communication alarm priority determines the priority level of object-generated alarms (0 – 10).

Retry attempts specifies the consecutive number of times Lookout attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the Aquatrol object generates an alarm and releases the communication port back to the communications service which then moves on to the next device in the polling queue (if any).

Receive timeout is the time delay Lookout uses in waiting for a response from a device before retrying the request.

The **Skip every…** setting instructs Lookout not to poll a device it has lost communication with on every scheduled poll. Instead, Lookout skips the device in the polling cycle accordingly. Once communications have been reestablished, the device is polled on its regular cycle.

RTU Configuration Dialog Box

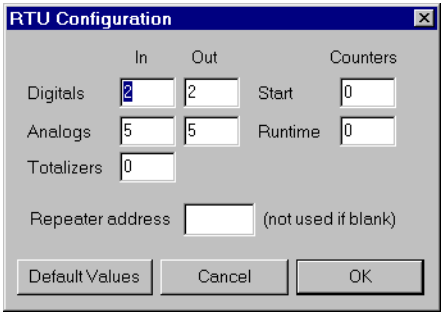


Figure 18-16. RTU Configuration Dialog Box

You must configure Lookout to match the configuration in your Aquatrol device.

Digital (In/Out) specifies configuration for the number of discrete I/O.

Analogs (In/Out) specifies configuration for the number of analog I/O.

Totalizers specifies configuration for the number of totalizers.

Start specifies configuration for the number of start counters.

Runtime specifies configuration for the number of runtime counters.

Repeater Address is the final destination address if a repeater is being used.

☞ **Note** ***The Aquatrol must be configured with all I/O grouped together by kind. Digital inputs must be the first physical inputs on the device, followed by digital outputs. Analog inputs are next, followed by analog outputs. If you do not configure the I/O this way, you can receive invalid data.***

Aquatrol Data Members

All readable and writable members (inputs/outputs), polling instructions, read/write blocking, serial port usage, and so on, are bundled with the object. Therefore, as soon as you create an Aquatrol object you immediately have access to all the object data members (see data member list below).

☞ **Note** ***Lookout protocol driver objects automatically generate an efficient read/write blocking scheme based on the inputs and outputs being used in your process file. You are not required to build your own I/O blocking table.***

Table 18-26. Aquatrol Data Members

Data Member	Type	Read	Write	Description
AI1-AI48	numeric	yes	no	Analog Input, 16 bit.
AO1-AO48	numeric	no	yes	Analog Output, 16 bit.
ColdRestart	logical	yes	no	True if the device power has cycled on/off or the reset button has been pushed.
CommFail	logical	yes	no	True if serial communications have failed.
ConfigError	logical	yes	no	True if the configuration of the device does not match the configuration set in the configuration dialog of the Aquatrol device.

Table 18-26. Aquatrol Data Members (Continued)

Data Member	Type	Read	Write	Description
DataFail	logical	yes	no	True if the device has had a data failure.
DI1-DI48	logical	yes	no	Discrete Input, 1 bit.
DO1-DO48	logical	no	yes	Discrete Output, 1 bit.
EEPROMerror	logical	yes	no	True if the device has an EEPROM error.
LowBattery	logical	yes	no	True if the device has a low battery.
PowerFail	logical	yes	no	True if there is no power to the device.
R1-R48	numeric	yes	no	Runtimes, 16 bit.
Poll	logical	no	yes	True initiates a Poll sequence.
PollRate	numeric	no	yes	Time interval of Polling sequence.
S1-S48	numeric	yes	no	Starts, 16 bit.
T1-T48	numeric	yes	no	Totalizers, 16 bit.
Update	logical	yes	no	False during Polling sequence.
WarmRestart	logical	yes	no	True if the device has been reset locally.

Aquatrol Status Messages

No response within timeout period
No response within timeout period for repeated message. Lookout did not received the expected response within the **Receive timeout** period. The object sent an inquiry and received an acknowledgment, but the device did not send an expected response to the request. This might happen if the response was interrupted. You may have to increase **Receive timeout**.

Frame Error (garbled): Invalid destination address
Lookout has received a frame with an invalid return address. Make sure that no duplicate addresses exist on the Aquatrol network.

Frame Error (garbled): Invalid source address
Lookout has received a frame from a device that you are not actively polling. Make sure that there is only one outstanding master request at a time.

Frame Error (garbled): Invalid message length for configuration
Lookout has received a frame whose length conflicts with the length expected based on the configuration settings in the Aquatrol device configuration dialog box.

Frame Error (garbled): Invalid CRC
Lookout has received a frame with an invalid CRC (cyclic redundancy check). Check for signal noise on Aquatrol network.

Invalid discrete address #: Check configuration settings
Invalid analog address #: Check configuration settings
Invalid runtime address #: Check configuration settings
Invalid start address #: Check configuration settings
Invalid totalizer address #: Check configuration settings
All the above errors mean that a I/O point being either read or written is out of range with respect to the configuration set in the Aquatrol device configuration dialog box.

ASCII

ASCII is a protocol driver class Lookout uses to communicate with any serial device that accepts ASCII characters. This object is only available with 32-bit versions of Lookout.

An ASCII object contains no predefined data points. When you create an ASCII object, you must define your data request strings as well as the template Lookout uses to parse the response frame.

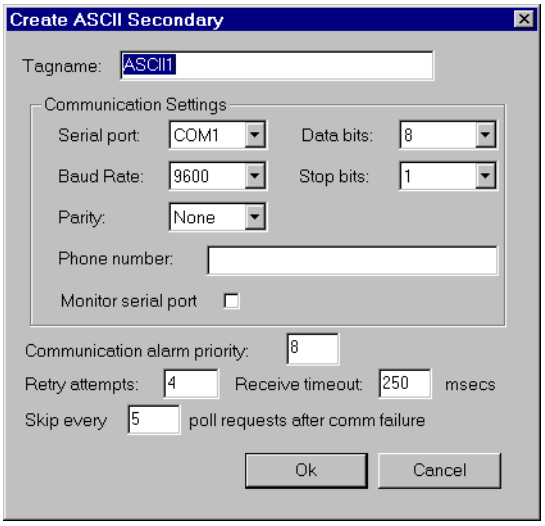


Figure 18-17. ASCII Definition Parameters Dialog Box

Serial port specifies which COM port the object uses for communicating to the external device. This does not specify the communication type. Communication type is determined by the **Options»Serial Ports…** command.

Baud rate indicates the rate that Lookout uses to communicate with the hardware device.

Data bits indicates the number of data bits that Lookout uses to communicate with the hardware setting should match the selection made on the physical device. device. This setting should match the selection made on the physical device.

Stop bits indicates the number of stop bits that Lookout uses to communicate with the hardware device. This setting should match the selection made on the physical device.

Parity indicates the parity that Lookout uses to communicate with the hardware device. This setting should match the selection made on the physical device.

Phone number specifies the number to be dialed if the selected serial port is configured for dial-up. This number only applies to the individual protocol object.

Monitor Serial Port specifies whether you can receive unsolicited frames.

Communication alarm priority determines the priority level of alarms generated by the ASCII object. Such alarms are typically related to communications with the physical device.

Retry attempts specifies the number of times Lookout attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the ASCII object generates an alarm and releases the communication port back to the communications service which then moves on to the next device in the polling queue (if any). Refer to Chapter 6, *Serial Communications*, for more information.

Receive timeout is the amount of time Lookout waits for a response from a device before retrying the request.

The **Skip every *N* poll requests after comm failure** setting instructs Lookout not to poll a device it has lost communication with on every scheduled poll. Instead, Lookout polls the device only once in the specified number of poll cycles. Once communication has been reestablished, the device is polled on its regular cycle.

ASCII Data Members

Table 18-27. ASCII Data Members

Data Member	Type	Read	Write	Description
RequestFormat	text	no	yes	Format used to create request frame.
ResponseFormat	text	no	yes	Format used to parse response frame.
Send	logical	no	yes	Sends request frame.

Table 18-27. ASCII Data Members (Continued)

Data Member	Type	Read	Write	Description
RQV1, RQV100	numeric	no	yes	Variable list used to populate request frame with numeric values.
RQV1.txt, RQV100.txt	text	no	yes	Variable list used to populate request frame with text values.
RQV1.logical, RQV100.logical	logical	no	yes	Variable list used to populate request frame with logical values.
RSFilter	text	no	yes	All characters in this string will be filtered out of the incoming response before processing.
RSSum1:1 - RSSum255:255	numeric	yes	no	Response byte sum
RQSum:1:1 - RQSum255:255	numeric	yes	no	Request byte sum
RSV1, RSV100	numeric	yes	no	Variable list used to store values retrieved from response frame.
RSV1.txt, RSV100.txt	text	yes	no	Variable list used to store values retrieved from response frame.
RSV1.logical, RSV100.logical	logical	yes	no	Variable list used to store values retrieved from response frame.
OffHook	logical	no	yes	Keeps the driver from releasing the serial port.
Request	text	yes	no	Exact request frame sent.
Response	text	yes	no	Exact response frame received.
CommFail	logical	yes	no	Object-generated signal that is on if Lookout cannot communicate with the device(s).
Update	logical	yes	no	Object-generated signal that pulses low each time it polls the device.
RSVn, RSVn.txt and RSVn.logical all represent the same value in different forms RQVn, RQVn.txt and RQVn.logical all represent the same value in different forms				

Request and Response Format Strings

The request and response format strings consist of static characters and markers that control how the request and response frames respectively are formatted or decoded. The request format string is used to **create** the request frame, which is sent to the device, while the response format string is used to **decode** the response frame, which comes from the device.

Static characters in the format strings are reproduced exactly in the request or response frame. Markers specify the location within the frame and type of data which should be found there, such as five characters read as an unsigned integer, for example. The ASCII object constructs a request frame by processing the sequence of static characters and markers in the request format string, and including data from RQV data members.

The response format string decodes a response frame using an analogous process, storing the results in RSV data members.

To construct a request frame, the ASCII object parses the request format string character by character. Static characters are copied directly to the request frame. When a marker is encountered the ASCII object reads a value from the appropriate RQV variable and places it into the request frame.

There are 100 RQV and RSV values provided for in the ASCII object data member collection. The first marker in a format string uses the value from RQV1 (or RQV1.txt or RQV1.logical), the next marker uses the value RQV2, and so on. Values taken from Response strings are stored in RSV data members in the same way.

Keep in mind that writing into RQV1 changes the value both for RQV1.text and RQV1.logical. Their only difference is the format in which they are represented. The same principle applies to the RSV data members.

☞ **Note** ***There is no precedence to the order in which multiple objects connected to the same variable number initialize upon opening the process file. Consider, for example, the case in which a Pot object is connected to RQV1 while a TextEntry object is connected to RQV1.txt. You should take care to initialize such variables to the proper value after opening a process file.***

To decode a response frame, the ASCII object compares the response frame to the response format string character by character. The static characters in the response frame must match those in the response format string or the decoding process terminates. Static characters are, in effect, discarded by

the ASCII object as they are matched between the response format string and the response frame.

When the ASCII object encounters a marker, it places the data indicated by the marker into the appropriate RSV data member.

The conversion of a portion of the response frame to a data type specified by a marker in the response format string must be valid, or the process will terminate.

If nothing halts the process, decoding terminates when the end of the response frame string is reached.

There are examples of both request frames and response frames at the end of this section, but for the examples to make sense, you must first understand the ASCII object markers.

ACSII Object Markers

The general format for a marker is:

%[width][type]

Each field in the marker format is a single character or a number signifying a particular format option.

The **%** sign denotes the beginning of the marker. If the percent sign is followed by a character that has no meaning as a format-control character, that character and the following characters (up to the next percent sign) are treated as static characters, that is, a sequence of characters that must match the frame exactly. For example, to specify that a percent-sign character is a static character part of the frame, use %%.

Width is a positive decimal integer specifying the number of characters that particular value occupies in the frame. By default ASCII pads the value with blank spaces if the value takes up fewer characters than the value specified by width. Including a 0 before the width value forces the ASCII object to pad with zeroes instead of blank spaces.

Type determines whether the field is interpreted as a character, a string, or a number.

Table 18-28. Data Types Allowed by ASCII

Character	Data Type
d	Decimal integer
x, X	Hexadecimal integer
u	Unsigned decimal integer
f	Floating-point
s	String
b	Byte (binary)

The simplest format specification contains only the percent sign and a type character (for example, %s). That would place the value in the response frame in the RSV1.txt data member.

Request Format String	RQV1	Request Frame
>%5d	34	> 34
>%05d	34	>00034

The request format string also has a precision value in the form **%[width].[precision][type]**. This specifies the number of digits to the right of the decimal point, if any, in the request frame. If you use a float (%f) and do not specify a precision value, the ASCII object assumes a default of 6.

Characters are converted and stored in RSV data members from response frames in the order they are encountered in the response format. However, fewer than **[width]** characters may be read if a white-space character (space, tab, or newline) or a character that cannot be converted according to the given format occurs before **[width]** is reached.

Values needed for request frames come from the RQV data members, and are also used in the order in which they occur in the request format.

To read strings not delimited by space characters, or that contain spaces, you can substitute a set of characters in brackets (**[ ]**) **s** (string) type character. The corresponding input field is read up to the first character that does not appear in the bracketed character set. Using a caret (^) as the first

character in the set reverses this effect: the ASCII object reads input field up to the first character that does appear in the rest of the character set

Response Format String	RSV1.txt	Response Frame
$[A – Z,a – z,]$	Natl Inst	$Natl Inst$
>[^,s]	days	>day

Notice that %[a – z] and %[z – a] are interpreted as equivalent to %[abcde…z], and that the character set is case sensitive.

☞ **Note** ***The brackets only work in response format strings. They have no effect in the request format string.***

The ASCII object scans each field in the response frame character by character. It may stop reading a particular field before it reaches a character for a variety of reasons:

- The specified width has been reached.
- The next character cannot be converted as specified.
- The next character conflicts with a character in the response format string that it is supposed to match.
- The next character fails to appear in a given character set.

No matter what the reason, when the ASCII object stops reading a field, the next field is considered to begin at the first unread character. The conflicting character, if there is one, is considered unread and is the first character of the next field.

Entering ASCII Object Format String

For a static connection to one of the format data members, enter your format string in the yellow field box in the Edit Connections dialog box. Remember to begin and end the format strings with quotation marks so that Lookout accepts the string input.

You can also connect any valid text data member, such as a text entry object, to the format data members.

Request Frame Construction Examples

Request Format String	RQV	Request Frame
<01%4u%s	RQV1=1234 RQV2.txt=Steph	<011234Steph
<01%04u%s	RQV1=34 RQV2.txt=Steph	<010034Steph
<01% 4u%s	RQV1=34 RQV2.txt=Steph	<01 34Steph

A zero in front of the four pads with zeroes; a space pads with spaces.

Response Format Examples

Response Frame	Response Format String	RSV
*(16.38:	*(%52f:	RSV1=16.38

Note ***The decimal point counts as a character when decoding floats (%f). Also, decimal points denoting precision are not allowed when decoding a float in the response frame.***

>>Test Text<<	>>%s<<	RSV1.txt=Test

The space between the words terminates the conversion. See the bracketed character example above in order to span a space or other special characters.

>>Test Text<<	>>%s%s<<	RSV1.txt=Test RSV2.txt=Text
>>DogCat<<	>>%3s%3s<<	RSV1.txt=Dog RSV2.txt=Cat

The response format uses a space as a delimiter.

Using Sum Data Members

The ASCII object includes summing data members you can use to calculate checksum characters. This can be a checksum you want to write into an outgoing request frame or a checksum you want to verify in an incoming response frame.

For example, if you want to calculate a checksum for the request `A00B`, you would use an `RQSum` (request sum) data member. In the case of `A00B`, you would use `RQSum1:4`, which would give you a sum of the ASCII byte values of characters 1 through 4. Once you have this sum, you can manipulate it mathematically any way necessary for the checksum value you need. You can then insert this value at the end of your frame as a byte (`%b`) or a series of bytes.

The same technique works in reverse for `RSSum` (response sum) data members.

For example, consider the response `Z00A@`. You know that you are expecting 4 bytes plus a checksum. Assuming that this checksum calculation involves the first four characters, use `RSSum1:4` to the byte sum of characters 1 through 4. After performing the appropriate mathematical manipulation, you can compare this value with the actual byte read from the frame, and determine when there is a checksum failure.

☞ **Note** ***There are many different methods for calculating checksums, and these data members cannot support all of them. Before attempting to use them for checksum calculation, make sure your checksum can be calculated from a simple byte sum of characters in the frame.***

ASCII Error Messages

No response from device within timeout period
Lookout received no response from the device within the **Receive timeout** period. The ASCII object was able to establish a socket, but when it sent a message to the device, the device did not respond—as if it were not there. You may have to significantly increase **Receive timeout** (and **Poll Rate**) to ensure Lookout is allowing enough time to receive the expected response. Also, verify your cable connections, power, configuration settings, and IP settings.

Not enough data to send a valid frame
This means that the ASCII object has not received enough data to fill in all the variables in the Request Format frame. This could mean that you do not have connections made to all of the RQVs that the ASCII object is expecting.

Frame Error (garbled)
ASCII got a response frame, but static characters in the response did not match up to the response format string.

Data type or length does not match format string
ASCII got a response frame, but certain characters in the response were not in the format stated by the markers in the response format string.

Related Objects ***IPASCII***

Average

Average actively calculates the average value of **Data** over time. Average is only active when the **Enable** expression is true and resets to zero when the **Reset** expression transitions from off to on. Average also maintains an array of up to 35 previous averaged values. If **Enable** is left blank, the object always actively calculates the average.

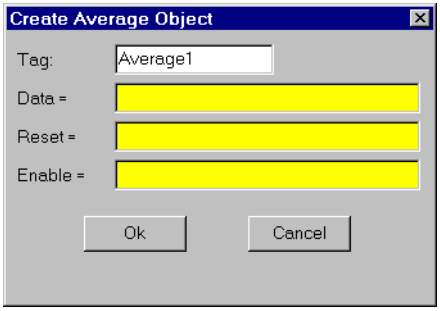


Figure 18-18. Average Definition Parameters Dialog Box

Data is a numeric expression while **Reset** and **Enable** are logical expressions.

☞ **Note** ***Average does not have a display parameters dialog box. However, you can easily display the result of its output signal by referencing it in an expression.***

Average Data Members

Table 18-29. Average Data Members

Data Members	Type	Read	Write	Description
(implicit)	numeric	yes	no	Current average calculated since the most recent Reset signal. Updated approximately once per second.
1 – 35	numeric	yes	no	Previous average values. Signal 1 is the most recent prior average since the Reset signal went high.
DataReset	logical	no	yes	Upon transition from FALSE to TRUE, resets to zero all data members—including the current average and all previous averages.

Comments ***The*** *Reset* ***expression could be a regular pulse interval created by a TimeOfxxxx timer, so that the pulse is synchronized to the top of the hour or day. For example, if you want to calculate the daily average flow rate, use the output signal from a TimeOfDay timer or a daily Spreadsheet object to reset the average calculation at the beginning of each day. If you want to calculate the average flow rate only when a pump is running, use the input signal from the pump motor relay in the*** *Enable* ***parameter.***

Related Objects ***Minimum, Maximum, Sample***

Counter

Counter counts the number of times that the **Count** expression transitions from off to on. The digital display shows the number of pulses counted so far, and is updated approximately once per second—however, it can receive and count multiple pulses within a given second. The counter can count to just under 4,503,600,000,000,000 or 142,710 years worth of pulses at one pulse per second. When the **Reset** expression transitions from off to on, the counter resets to zero.

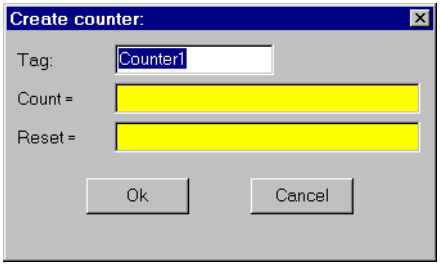


Figure 18-19. Counter Definition Parameters Dialog Box

Both **Count** and **Reset** are logical expressions.

☞ **Note** ***Counter does not have a display parameters dialog box. However, you can easily display the result of Counter output signal by referencing its data member in an expression.***

Counter Data Members

Table 18-30. Counter Data Members

Data Member	Type	Read	Write	Description
(implicit)	numeric	yes	no	numeric total of pulse count

Comments ***You should not use Counters to count external pulse signals that cycle more often than about once per second. For higher counting speeds, use the accumulator capabilities built into your PLC.***

Related Objects ***Accumulator, Integral***

Cutler-Hammer

Cutler-Hammer is a protocol driver class Lookout uses to communicate with Cutler-Hammer devices using serial communications.

It supports reading and writing of all predefined data points allowed by the particular Cutler-Hammer model. When you create a Cutler-Hammer object, you have immediate access to all the object data members. See the *Cutler-Hammer Data Member Set* table for more information on data members for this object.

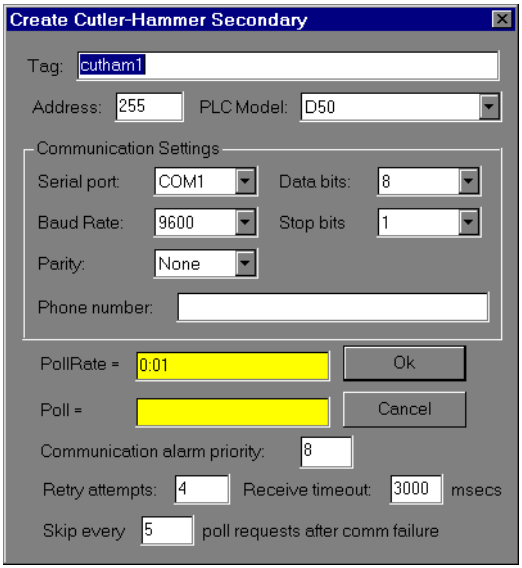


Figure 18-20. Cutler-Hammer Definition Parameters Dialog Box

Address specifies which Cutler-Hammer device you are communicating with using this object. This number is between 0 and 191, and is set on the device. (An address number of 255 broadcasts to all devices connected to the network.)

PLC Model specifies what model of Cutler-Hammer device you are using. The only model supported now is the Cutler-Hammer D50.

Serial port specifies which comm port the object uses for communicating to the external device. This does not specify the communication type. Communication type is determined by the **Options»Serial Ports** command.

☞ **Note** ***For Lookout to communicate correctly with the Cutler-Hammer D50, you must configure your COM port as*** `radio` ***using the*** **Options»Serial Ports** ***command.***

Data rate indicates the baud rate that Lookout uses to communicate with the hardware device. This setting should match the selection made on the physical device.

Data bits indicates the number of data bits that Lookout uses to communicate with the hardware device. This setting should match the selection made on the physical device.

Stop bits indicates the number of stop bits that Lookout uses to communicate with the hardware device. This setting should match the selection made on the physical device.

Parity indicates the parity that Lookout uses to communicate with the hardware device. This setting should match the selection made on the physical device.

Phone number specifies the number to be dialed if the selected serial port is configured for dial-up. This number only applies to the individual protocol object.

PollRate is a numeric expression that determines how often to poll the device. Lookout converts the numeric value of **PollRate** into a time signal that represents days and fractions of a day. The object then polls the device at the specified time interval. Normally, this is a time constant such as 0:01 (one second). See the *Numeric Data Members* section in Chapter 2, *Introduction*, for information on entering time constants.

Poll is a logical expression. When this expression changes from false to true, Lookout polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Communication alarm priority determines the priority level of alarms generated by the Cutler-Hammer object. Such alarms are typically related to communications with the physical device.

Retry attempts specifies the consecutive number of times Lookout attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the Cutler-Hammer object generates an alarm and releases the comm port. See Chapter 6, *Serial Communications,* for more information.

Receive timeout is the time delay Lookout uses in waiting for a response from a device before retrying the request.

The **Skip every *N* poll requests after comm failure** setting instructs Lookout to not poll a device it has lost communication with on every scheduled poll. Instead, Lookout skips the device in the polling cycle accordingly. Once communications have been reestablished, the device is polled on its regular cycle.

Cutler-Hammer Data Members

As with all Lookout drivers, you can access I/O points and other data through data members. The following is a table of data members currently supported by the Cutler-Hammer object class.

Table 18-31. Cutler-Hammer Data Member Set

Data Member	Type	Read	Write	Description
R00000.logical - R02915.logical	logical	yes	yes	External inputs and outputs, and special function registers
M00000.logical - M03115.logical	logical	yes	yes	Internal relay
M0000 - M0031	numeric	yes	yes	Internal relay
K00000.logical - K01515.logical	logical	yes	yes	Keep relay
K0000 - K0015	numeric	yes	yes	Keep relay
W0000 - W2815	numeric	yes	yes	Timers, counters, and word registers
Update	logical	yes	no	Goes high when Lookout begins a poll cycle on the device
CommFail	logical	yes	no	Goes high if Lookout cannot communicate with the device
Poll	logical	no	yes	When transitioned from low to high, Lookout begins a poll cycle on the device

Table 18-31. Cutler-Hammer Data Member Set (Continued)

Data Member	Type	Read	Write	Description
PollRate	numeric	no	yes	Specifies the frequency at which Lookout polls the device

For a more complete definition of the function of these data members, see Cutler-Hammer documentation.

Not all of these data members are valid for every Cutler-Hammer device. See Cutler-Hammer documentation to see which data members are valid for your particular model number.

Cutler-Hammer Status Messages

No response within timeout period
Lookout received no response from a device within the **Receive timeout** period. The Cutler-Hammer object is able to use the comm port, but when it polls the device, it does not respond—as if it is not even there. If you have daisy-chained several devices, you may have introduced an inherent delay. You may have to significantly increase **Receive timeout** (and **Poll Rate**) to ensure Lookout is allowing enough time to receive the expected response. This increase has nothing to do with the processing capabilities of Lookout. Rather it is based solely on **Data rate** and the number of devices on the chain. Also, verify your baud rate settings, cable connections, power, configuration settings, comm port settings, and polling addresses.

Invalid query acknowledge frame
This means that the query acknowledge frame sent from the PLC in response to the command sent by Lookout out was invalid.

Invalid PLC address in response frame
This means that the address within the response frame from the PLC does not match exactly the address in the command sent out by Lookout. You may possibly be requesting an address that is outside the valid range from Cutler-Hammer devices.

Invalid CRC in response
This means the checksum (CRC in this case) failed in a frame received by Lookout. Check cabling or for two or more devices with the same address.

Garbled or invalid frame
Frame received without format characters in their proper positions. Check the **Receive gap** setting.

Invalid response frame
This means that the response frame from the PLC did not have the expected number of bytes in it.

Invalid function code in response frame
This means that the function code within the response frame from the PLC does not match the function code in the command sent out by Lookout.

Wrong query, length, or designation
This is an error returned as an error code frame from the Cutler-Hammer device. This means that the device could not successfully interpret the query you just sent it.

Designated range exceeded
This is an error returned as an error code frame from the Cutler-Hammer device. It means you have requested an address beyond the valid range of addresses.

Command too long in length
This is an error returned as an error code frame from the Cutler-Hammer device. This means that the frame just sent has exceeded the maximum amount information that can be transported in a single frame. (256 bytes)

Abnormal communications command
This is an error returned as an error code frame from the Cutler-Hammer device.

Cutler-Hammer models supported:
D50

DataTable

DataTables are used in the following applications:

- Multiplexing various data sources into a single display template.
- Importing or exporting large quantities of data to other applications using DDE. (Unlike the DdeTable, the DataTable provides bidirectional DDE communications. See *DdeTable*.)
- Networking multiple Lookout nodes (see Chapter 14, *Networking*).

A DataTable contains a matrix of rows and columns, much like a spreadsheet. Each column is represented by a letter (A – IV). Each row is represented by a number (1 – 1,000). Letter-number combinations represent intersections of rows and columns. Such intersections are called cells. Any given cell contains a value. A cell value can be numeric, logical, or textual.

Note ***DataTables are advanced tools that require a mastery of Lookout object databases. Make sure you understand editing, connecting to, and using object databases and aliases before creating a DataTable object. See Chapter 5,*** *Developer Tour****, for more information on object databases.***

The following dialog box appears when you create a DataTable:

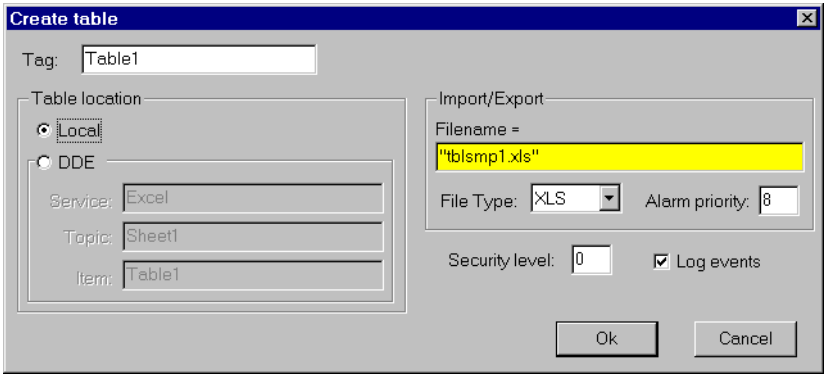


Figure 18-21. DataTable Definition Parameters Dialog Box

Specify **DDE** parameters if you want to use the DataTable to *import* large quantities of data from an external source. Typical external sources include another DataTable within a networked Lookout application or an array of spreadsheet data. See Chapter 14, *Networking*, for more information on networking.

Specify **Local** if you are using the DataTable to *export* large quantities of data from Lookout. Also specify **Local** if you are using the DataTable to multiplex various signals into a single display or graphic template.

The export and import data members control the transfer of data between Lookout and a Microsoft Excel spreadsheet. The spreadsheet filename must be entered in the **Import/Export** section of the **Create table** dialog box.

☞ **Note** ***The data table import and export data members only work with Microsoft Excel Version 4 at this time. This feature does not work with Excel Version 5 or greater.***

Instead of entering a file name in the **Import/Export** dialog box, you can enter a Lookout expression in the **Filename** field. You can then import and export different files using a switch setting, a text entry box, or some other expression input. If something goes wrong with the transfer of data, including data corruption, Lookout reports an alarm.

Multiplexing Displays and Graphics

You can use DataTables to multiplex signals into a single control panel, used as a template. For instance, assume you have nine identical pump stations. At each site you have a single PLC monitoring the pressure, flow rate, and status of two pumps. You are also controlling an analog output with an operator setpoint from Lookout. Instead of developing nine identical control panels in Lookout, you can build just one panel and multiplex the signals from the nine sites into a single set of graphics.

The following figure is a graphical representation of the connections for a possible DataTable.

	A	B	C	D	E	F
1	Site1	PLC1.press	PLC1.flow		PLC1.pmp1	PLC1.pmp2
2	Site2	PLC2.press	PLC2.flow	Setpoint Value Column	PLC2.pmp1	PLC2.pmp2
3	Site3	PLC3.press	PLC3.flow		PLC3.pmp1	PLC3.pmp2
4	Site4	PLC4.press	PLC4.flow		PLC4.pmp1	PLC4.pmp2
5	Site5	PLC5.press	PLC5.flow		PLC5.pmp1	PLC5.pmp2
6	Site6	PLC6.press	PLC6.flow		PLC6.pmp1	PLC6.pmp2
7	Site7	PLC7.press	PLC7.flow		PLC7.pmp1	PLC7.pmp2
8	Site8	PLC8.press	PLC8.flow		PLC8.pmp1	PLC8.pmp2
9	Site9	PLC9.press	PLC9.flow		PLC9.pmp1	PLC9.pmp2

Figure 18-22. Graphical Representation of a DataTable Showing Connections

☞ **Note** ***At this time there is no actual view in Lookout for you to see this table. It is a representation, not a screen capture of a Lookout dialog box or display element.***

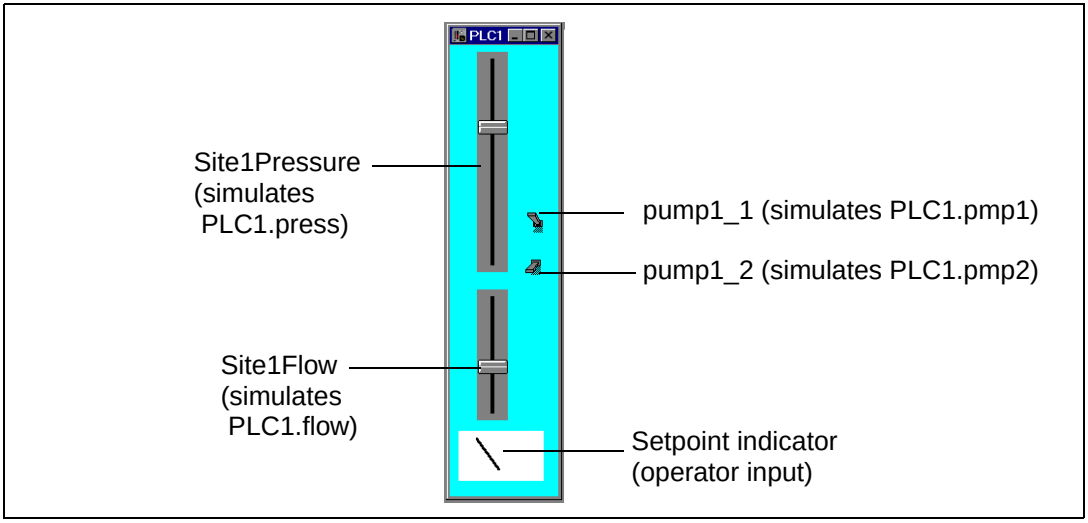
Each row of the table is connected to the site name, the data you want to keep track of from the PLC at that site, and a place for the operator setpoint data for that site.

Notice the tagnames of the nine PLCs (`PLC1, PLC2…`). Each value has an alias (`press`, `flow`, `pmp1`, `pmp2`). Each cell represents a connection made from that PLC output to the that cell of the DataTable.

The open column D represents the connection of a single Pot object called `Setpoint` to the column—not to individual cells. This way Lookout only changes the value of a cell in column D when that particular row has been made active. You connect the individual cells in column D (D1 through D9) to the correct holding registers (outputs) on the respective PLCs. For example, `Table1.D1` would be connected to the appropriate data member of `PLC1` – `Table1.D2` to the appropriate data member of `PLC2`, and so on. In other words, when you intend to multiplex signals to a panel through a data table connect inputs from a PLC or RTU to individual cells. Connect operator setpoints (outputs) to columns.

DataTable Example

To practice using a DataTable, you can create a PLC simulator as shown in the following illustration. You can use a separate control panel for each PLC output simulator, or place all the objects you make on one panel.



Use two slider pots as your pressure and flow outputs, and two switches as the on/off indicators for pumps one and two. You can use the dial gauge at

the bottom to check the operator input, `Setpoint`. Instead of connecting the cells of your data table to PLC outputs, you connect directly to the Lookout objects, named after the outputs for clarity.

You should create at least three sets of PLC input/output simulators, The following example refers to the 9 PLCs mentioned in the *Multiplexing Displays and Graphics* section, but you can explore using a data table using 2 or 3 simulators.

Connect specific DataTable cells from the `D` column to the Setpoint indicator for each PLC simulator.

After you have built your simulators, open a new Control Panel to use as your display and create a DataTable. You will create the rest of the display in a later step.

When it first appears, the DataTable contains the number 0. This indicates the contents of cell A1. You can increase the size of the display window, but you cannot show the entire array of data in the table. You can view the contents of any cell in column A by clicking on the window when you are in run mode. Selecting the contents of that cell activates that row of the data table.

Enter text expressions into the cells of column A to act as your table index. For example, enter the string `"Site 1"`as the connection to your DataTable `A1.txt` data member. Connect the outputs of your PLC simulators to the cells in the B, C, E, and F columns discussed below. You will connect an operator input to the entire column D, and than later connect individual cells in the D column to your PLC simulators.

	A	B	C	D	E	F
1	Site 1	Site1Pressure	Site1Flow		pump1_1	pump1_2
2	Site 2	Site2Pressure	Site2Flow		pump2_1	pump2_2
3	Site 3	Site3Pressure	Site3Flow		pump3_1	pump3_2
4	Site 4	Site4Pressure	Site4Flow		pump4_1	pump4_2
5	Site 5	Site5Pressure	Site5Flow		pump5_1	pump5_2
6	Site 6	Site6Pressure	Site6Flow		pump6_1	pump6_2
7	Site 7	Site7Pressure	Site7Flow		pump7_1	pump7_2
8	Site 8	Site8Pressure	Site8Flow		pump8_1	pump8_2
9	Site 9	Site9Pressure	Site9Flow		pump9_1	pump9_2

Connecting Signals to DataTables

To connect a value to a particular cell or column within a DataTable, use the **Object»Edit Connections…** command. Select the specific data member of the DataTable to be written, and identify the data member of the source object.

Connecting to Cells

The value from `Site1Pressure` is numeric. To write the value of `Site1Pressure` into cell `B1`, connect the simulated output to `B1`—not `B1.logical` or `B1.txt`. The **Edit connection** dialog box is shown in the following illustration.

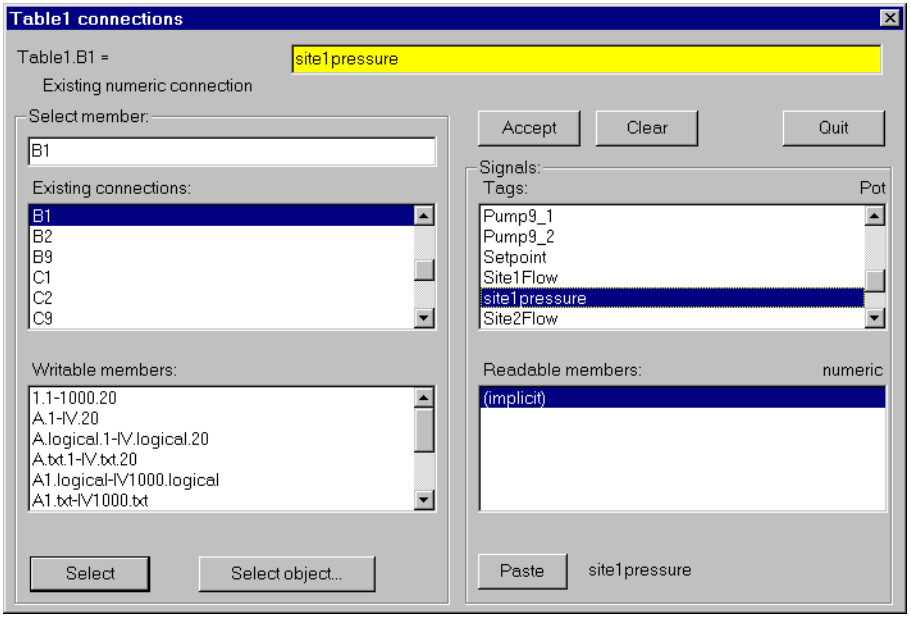


Figure 18-23. Edit Connections Dialog Box

After you establish a connection to a cell, the value within the cell changes any time the expression changes. Any time the value of `Site1Pressure` changes, the value within cell `B1` changes.

Connecting to Columns

To connect the value of the `Setpoint` potentiometer to column D, select `D`—not `D.logical` or `D.txt`—as shown in the following illustration. As with the previous value, you use `D` because you are using

a numerical value. There is no number following the D because you are connecting to a column, not a cell.

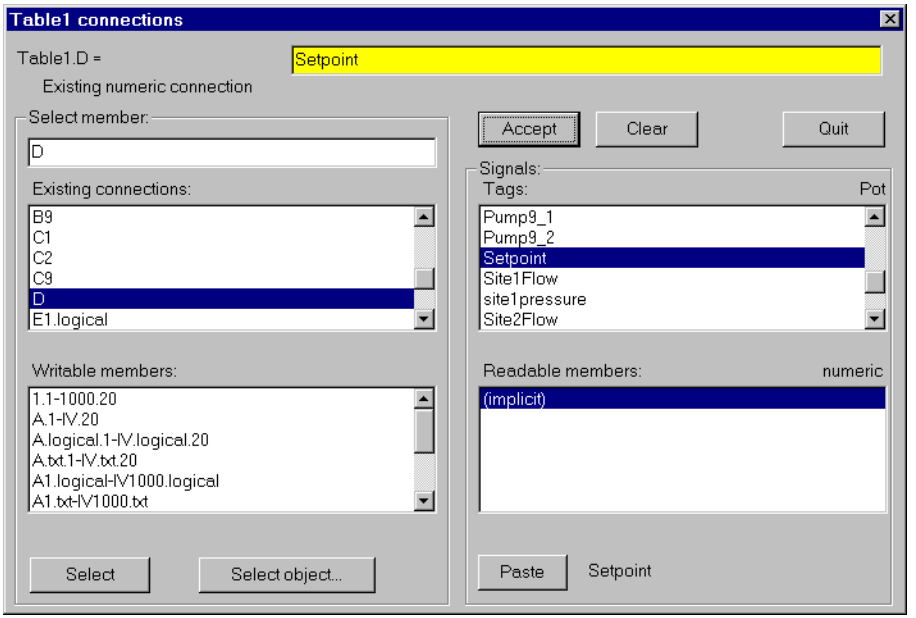
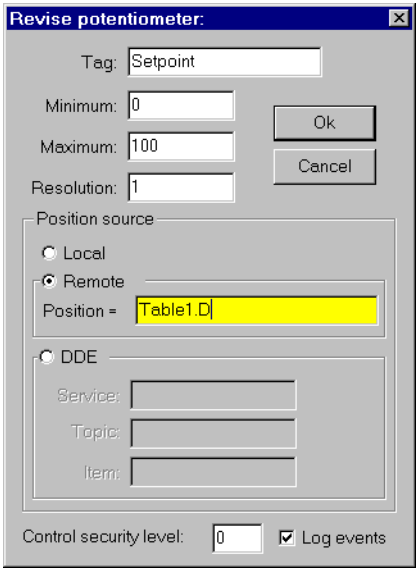


Figure 18-24. Edit Connections Dialog Box

Once a connection to a column is established, the value within the cell *at the currently selected row* changes when the expression changes. So, if you activate row 4 and change the value of `Setpoint` changes, the value within cell `D4` changes. No other cells are affected until you move the cursor to activate another row and the operator adjusts the pot.

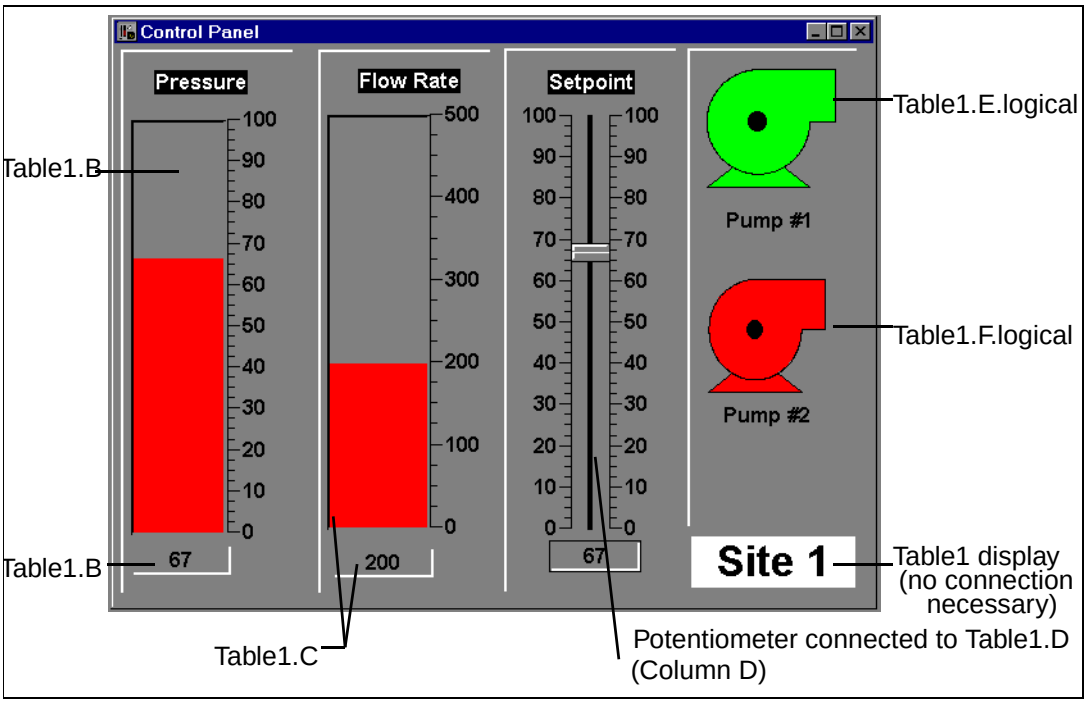
Reading a Cell Value Back to a Lookout Object

When you connect the display panel potentiometer called `Setpoint` to `Table1.D`, you should configure your pot with the appropriate **Remote** parameter to ensure that it automatically adjusts to track the value in any cell (within the specified column) when the cursor moves to a new row.



The Display Panel

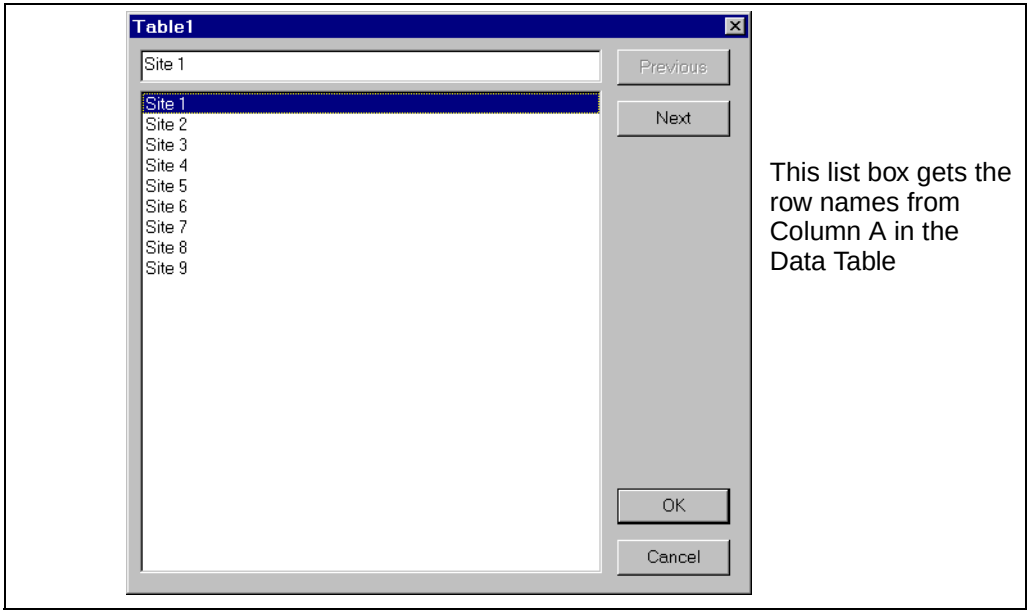
Now that your DataTable (called `Table1`) is complete, you can return to your display panel (where you placed the DataTable object initially) and begin to build a single control panel that allows you to multiplex the data from your PLCs. A sample panel is shown in the following illustration.



Instead of using expressions that reference the actual values from your driver objects to display values, use the column names from `Table1`. For instance, the bar graph and numeric readout for Pressure both represent the expression `Table1.B`. The actual value for `Table1.B` depends on the what row is currently active in the DataTable. In the illustration row 1 is active, so all the expressions return the value in their respective columns at row 4. The callouts in the picture indicate how the control panel was developed. All are Lookout expressions except the Pot and the DataTable, which are displayable objects.

Operating Your Multiplexed Panel

The plate in bottom right corner of the control panel is the DataTable object (`Table1`). To view a different site with the control panel (that is, to activate a different row, thereby selecting a different PLC), click on `Table1` and the following dialog box should appear.



By selecting a site, you connect your control panel to the PLC at that site. This is referred to as moving the cursor to that row.

DataTable Cursors

The cursor is a DataTable pointer that you can move from row to row to activate the values in the cells of that row. There are several methods for controlling the location of the cursor:

- Connect a numeric expression to the `cursor` data member. A typical example would be to connect a potentiometer (minimum = 1, maximum = the number of rows in table, resolution = 1) to the cursor.
- Connect logical expressions to appropriate row numbers. A typical example would be to create a pushbutton for every row and then connect them to their respective row numbers.
- Use the display (list box) built into the DataTable object. A typical example was described above where you connected text values to cells in column A and then displayed the table as a plate.

The following example provides a graphical representation of a DataTable (called `Table1`) showing typical values within its many cells. You can create a multiplex effect in Lookout by referencing column names and then selecting the row with the information you want to use. If you reference column names (instead of individual cells), the DataTable outputs only the

values within the currently selected row. If you reference individual cells, the DataTable outputs the current value within the cell—no matter where the cursor is.

	A	B	C	D	E	F
1	Site 1	67.8	540	56	0	1
2	Site 2	77.9	460	43	0	1
3	Site 3	57.3	480	78	0	1
4	Site 4	57.8	410	51	1	1
5	Site 5	51.8	560	92	1	0
6	Site 6	88.3	490	40	0	1
7	Site 7	79.4	530	63	1	0
8	Site 8	92.1	520	71	1	0
9	Site 9	59.9	550	62	0	1

Outputs =	**Site2**	**77.9**	**460**	**78**	**0**	**1**

Figure 18-25. DataTable with Cursor at Row 2 and Corresponding Outputs

The value of cell B2 is currently 77.9. Therefore, the `Table1.B` data member is also 77.9. If you move the cursor to row 9, the value of `Table1.B` changes to 59.9, as shown below.

	A	B	C	D	E	F
1	Site 1	67.8	540	56	0	1
2	Site 2	77.9	460	43	0	1
3	Site 3	57.3	480	78	0	1
4	Site 4	57.8	410	51	1	1
5	Site 5	51.8	560	92	1	0
6	Site 6	88.3	490	40	0	1
7	Site 7	79.4	530	63	1	0
8	Site 8	92.1	520	71	1	0
9	Site 9	59.9	550	62	0	1

Outputs =	**Site9**	**59.9**	**550**	**62**	**0**	**1**

Figure 18-26. DataTable with Cursor at Row 9 and Corresponding Outputs

Using Multiple Cursors

The above description assumes you are using just one cursor. But a DataTable can have up to 20 cursors. Multiple cursors allow you to select multiple rows at the same time. When using multiple cursors, you also use multiple names for each column. For a given column, each name is associated with a given cursor. If you are using two cursors in the above

example (`cursor` and `cursor.2`), you can reference the column name of a given column as follows:

Table 18-32. Column Names

	A	B	C	D	E	F
cursor	A.txt	B	C	D	E.logical	F.logical
cursor.2	A.txt.2	B.2	C.2	D.2	E.logical.2	F.logical.2

Earlier, when you were using just one cursor, you connected the value of a potentiometer called `Setpoint` to column D. Subsequently the value of `Setpoint` was written to the cell at the row selected by the cursor. But when there are multiple cursors, you have to select which cursor you are writing to. Thus, depending on how you want your table to work, you might connect the potentiometer to both `Table1.D` and `Table1.D.2`.

DataTable Data Members

Table 18-33. DataTable Data Members

Data Members	Type	Read	Write	Description
(implicit)	DdeTable	no	no	Not displayable in Lookout, but it can be referenced by a DDE link from another application.
A1 – IV16384	numeric	yes	yes	Specified cell interpreted as numeric value
A1.logical –IV16384.logical	logical	yes	yes	Specified cell interpreted as logical value
A1.txt –IV16384.txt	text	yes	yes	Specified cell interpreted as text value
Cursor.1 – Cursor.20	numeric	yes	yes	Specifies the currently selected row of the indicated cursor.
1.1–1000.20	logical	no	yes	Specifies row (1, 2, 3, …1000) or specifies row.cursor (for example, 24.2 is the selector for row 24, cursor 2). Upon transition from false to true, the specified cursor moves to specified row.

Table 18-33. DataTable Data Members (Continued)

Data Members	Type	Read	Write	Description
A.1–IV.20	numeric	yes	yes	Specifies column names (for example, A, B, C…IV) or specifies column names and associated cursor numbers (such as, A.1, B.1, A.2, B.2, and so on.) *Read*—returns a numeric value from the cell specified by the column and currently selected row of the indicated cursor. *Write*—writes a numeric value into the cell specified by the column and currently selected row of the indicated cursor.
A.logical.1 – IV.logical.20	logical	yes	yes	Specifies column names (for example, A.logical, B.logical, C.logical, …IV.logical) or specifies column names and associated cursor numbers (such as, A.logical.1, B.logical.1, A.logical.2, B.logical.2, and so on.) *Read*—returns a logical value from the cell specified by the column and currently selected row of the indicated cursor. *Write*—writes a logical value into the cell specified by the column and currently selected row of the indicated cursor.

Table 18-33. DataTable Data Members (Continued)

Data Members	Type	Read	Write	Description
A.txt.1 – IV.txt.20	text	yes	yes	Specifies column names (for example, A.txt, B. txt, C. txt, …IV. txt) or specifies column names and associated cursor numbers (such as, A. txt.1, B. txt.1, A. txt.2, B. txt.2, and so on.) *Read*—returns a text value from the cell specified by the column and currently selected row of the indicated cursor. *Write*—writes a text value into the cell specified by the column and currently selected row of the indicated cursor.
enable	logical	yes	yes	If TRUE (the default), enables DDE. If FALSE, disables DDE. The default value is on. The input is ignored for non-DDE TextEntry objects.
export	logical	no	yes	When this input transitions from false to true, the Lookout data table is exported to the designated spreadsheet file.
import	logical	no	yes	When this input transitions from false to true, the Lookout data table imports data from the designated spreadsheet.
Update	logical	yes	no	Pulses each time the cursor changes rows. Often used to “call up” a control panel.

Related Objects ***DdeTable, DdeLink***

DdeLink

DdeLink creates a unidirectional Dynamic Data Exchange (DDE) link to another application. The other application could be running on the same computer or on another computer over a network. DdeLink objects provide an easy way to *import* remote values into Lookout. See Chapter 13, *Dynamic Data Exchange*, for more information on DDE. For each DdeLink object you define, Lookout creates a separate link to the other application. If you need to import large quantities of data, you should use the DdeTable or DataTable object. See *DdeTable* and *DataTable* for more information.

Note ***DdeLink does not have a display parameters dialog box. You can easily display the result of a DdeLink output signal by referencing its data member in an expression.***

DdeLinks on Same Computer

If you are importing values from another application running on the same computer, your DDE parameters will look similar to the ones below.

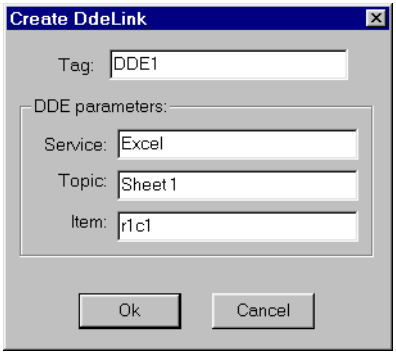


Figure 18-27. DdeLink Definition Parameters Dialog Box (Same Computer)

Service specifies the application name, **Topic** specifies the file, and **Item** points to the individual value (r1c1 refers to the cell at row1, column1. Unfortunately, Excel does not support the more convenient A1 cell references with DDE).

DdeLinks to Remote Computer

If you are importing values from another application running on a remote computer, your DDE parameters will look similar to the ones below.

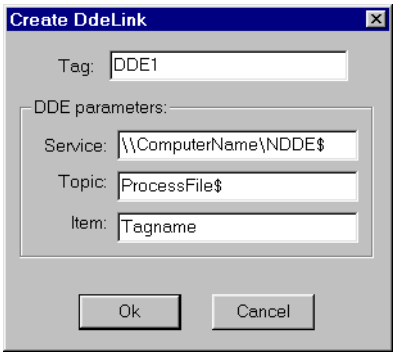


Figure 18-28. DdeLink Definition Parameters Dialog Box (Remote Computer)

Notice the differences in the **Service**, **Topic**, and **Item** parameters. The backslashes (\\) and dollar signs ($) are standard requirements for making network connections in Microsoft Windows for Workgroups. `ComputerName` specifies the network name of the computer you are connecting to. If you are connecting to a value in another Lookout application, `ProcessFile` is the Lookout file name running on the remote computer, and `Tagname` refers to the tag you are linking to.

DDELink Data Members

Table 18-34. DdeLink Data Members

Data Members	Type	Read	Write	Description
(implicit)	numeric	yes	no	DDE link interpreted as numeric value
enable	logical	yes	yes	If TRUE (the default), enables DDE. If FALSE, disables DDE. This input is ignored for non-DDE TextEntry objects.
logical	logical	yes	no	DDE link interpreted as logical value
txt	text	yes	no	DDE link interpreted as text value
hot	logical	yes	no	Status of DDE link

Related Objects ***DdeTable, DataTable***

DdeTable

DdeTable creates a unidirectional Dynamic Data Exchange (DDE) link to another application. The other application could be running on the same computer or on another computer over a network. See Chapter 13, *Dynamic Data Exchange*, for more information on DDE. You can use DdeTable to *import* large quantities of data from other applications. The table format is much more efficient at transferring data than the Link format because a table can contain hundreds or even thousands of data points that all share a single link. On the other hand, the link format can only transfer a single value per link—and every link requires a certain amount of CPU overhead. If you are only importing a relatively small amount of data, you may find the DdeLink technique easier to implement.

☞ **Note** ***DdeTable does not have a display parameters dialog box. You can display the result of a DdeTable output signals by referencing its data members in expressions.***

DdeTable on Same Computer

If you are importing values from another application running on the same computer, your DDE parameters will look similar to the ones below.

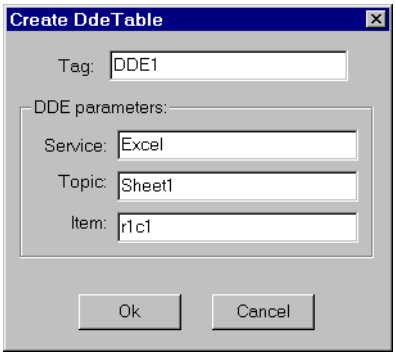


Figure 18-29. DdeTable Definition Parameters Dialog Box (Same Computer)

Service specifies the application name, **Topic** typically specifies the file, and **Item** specifies the particular data item name.

The example below shows an Excel spreadsheet with the highlighted range named Data.

Data | Pressures

	A	B	C	D
1	Pressures	Temps	Flows	Times
2	99	574.9	12	1:23:04
3	11	442	21	0:30:13
4	98.51	602.4	34	0:57:45
5				
6	Ammonia	Chlorine	Alum	Status
7	0.076	1.34	27.87	ON

You can now display any value in the range Data with the Lookout DdeTable object created above.

Select **Insert»Expression...** and then choose the DdeTable data member corresponding to the Excel spreadsheet cell containing the value you want to display. Make sure that the type of data member you select matches the type data in the spreadsheet cell.

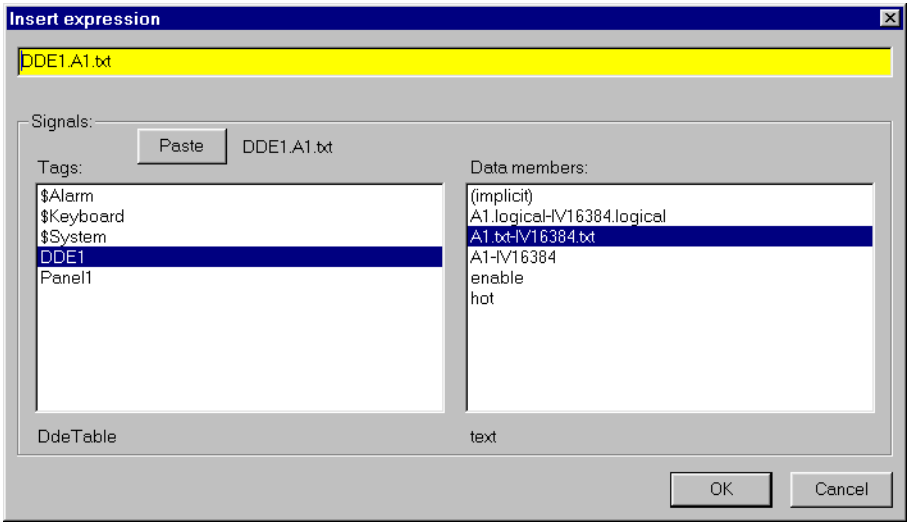
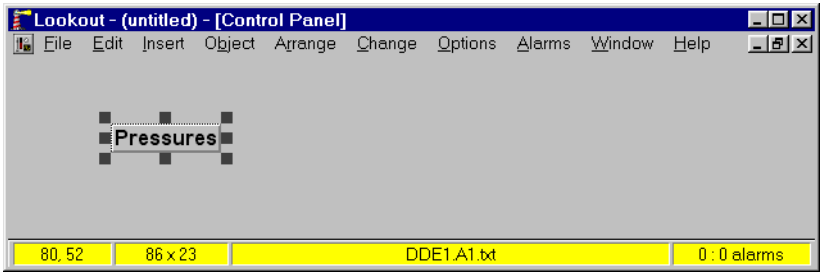


Figure 18-30. Inserting a DdeTable Expression

Notice the reference for the displayed item in the Lookout status bar.



DdeTable to Remote Computer

Using the DdeTable object over a network is somewhat different from the example above.

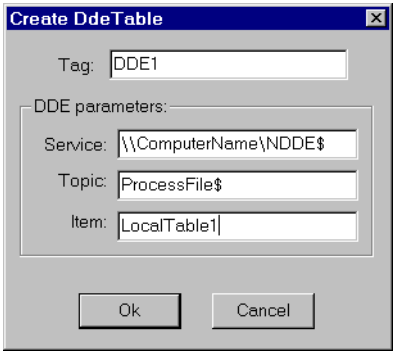


Figure 18-31. DdeTable Definition Parameters Dialog Box (Remote Computer)

Notice the differences in the **Service**, **Topic**, and **Item** parameters. The backslashes (\\) and dollar signs ($) are standard requirements for making network connections in Microsoft Windows for Workgroups. `ComputerName` specifies the network name of the computer you are connecting to. If you are connecting to a DdeTable or DataTable in another Lookout application, `ProcessFile` is the Lookout file name running on the remote computer, and `LocalTable1` refers to the DdeTable or DataTable object you are linking to.

DDETable Data Members

Table 18-35. DdeTable Data Members

Data Members	Type	Read	Write	Description
(implicit)	numeric	yes	no	Cell A1 interpreted as a numeric value
A1 – IV16384	numeric	yes	no	Specified cell interpreted as a numeric value
A1.logical – IV16384.logical	logical	yes	no	Specified cell interpreted as a logical value
A1.txt – IV16384.txt	text	yes	no	Specified cell interpreted as a text value
enable	logical	yes	yes	If TRUE (the default), enables DDE. If FALSE, disables DDE. This input is ignored for non-DDE TextEntry objects.
hot	logical	yes	no	Status of DDE link

Related Objects ***DataTable, DdeLink***

DelayOff

DelayOff is an adjustable delay timer. When **On/off signal** transitions to off, the **Timer delay** begins to count down. At the end of the delay countdown, the output signal turns off. **On/off signal** must remain off during the time delay period for the output signal to turn off. The output immediately turns on when the **On/off signal** turns on.

Timer delay can range from 0.0 seconds to several years, and the effective resolution is 0.1 seconds over the entire range. The timer display digitally shows the time delay remaining, and is updated approximately once per second. If the **On/off signal** is high, the timer display shows `on`. If the **Timer delay** period has expired, the display shows `off`.

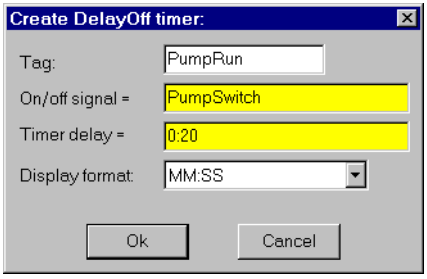


Figure 18-32. DelayOff Definition Parameters Dialog Box

The **On/off signal** is a logical expression while **Timer delay** is a numeric expression. Normally, this is a simple time constant such as 0:20—twenty seconds. See *Numeric Data Members* in Chapter 5, *Developer Tour*, for information on entering time constants.

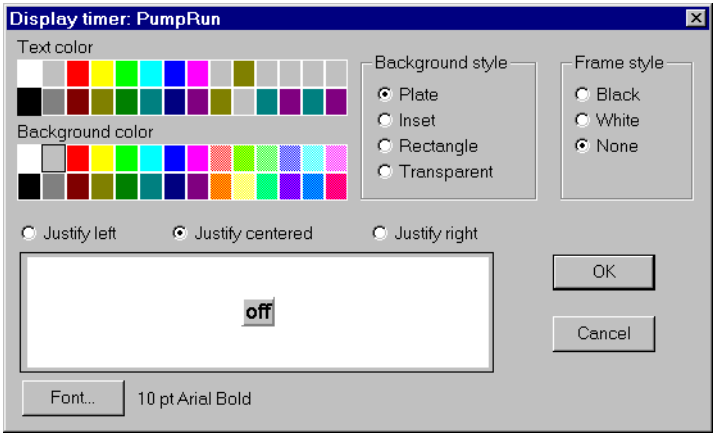


Figure 18-33. DelayOff Display Parameters Dialog Box

DelayOff Data Members

Table 18-36. DelayOff Data Members

Data Member	Type	Read	Write	Description
(implicit)	logical	yes	no	Logical timer value

Comments ***The DelayOff timer can prevent a pump from short-cycling.***

Related Objects ***DelayOn, Interval, OneShot, Pulse, TimeOf***

DelayOn

DelayOn is an adjustable delay timer. When **On/off signal** transitions to on, the **Timer delay** begins to count down. At the end of the delay countdown, the output signal turns on. **On/off signal** must remain on during the time delay period for the output signal to turn on. The output immediately turns off when the **On/off signal** turns off.

Timer delay can range from 0.0 seconds to several years, and the effective resolution is 0.1 seconds over the entire range. The timer display digitally shows the time delay remaining and is updated approximately once per second. The timer display shows `off` when the **On/off signal** is low. If the **Timer delay** period has expired, the display shows `on`.

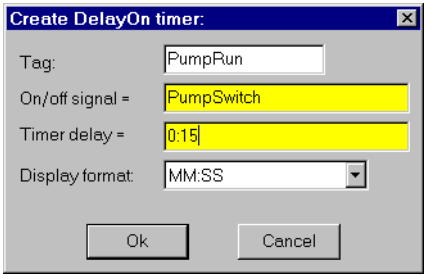


Figure 18-34. DelayOn Definition Parameters Dialog Box

The **On/off signal** is a logical expression while **Timer delay** is a numeric expression. Normally, this is a simple time constant such as 0:20—twenty seconds. See *Numeric Data Members* in Chapter 5, *Developer Tour*, for information on entering time constants.

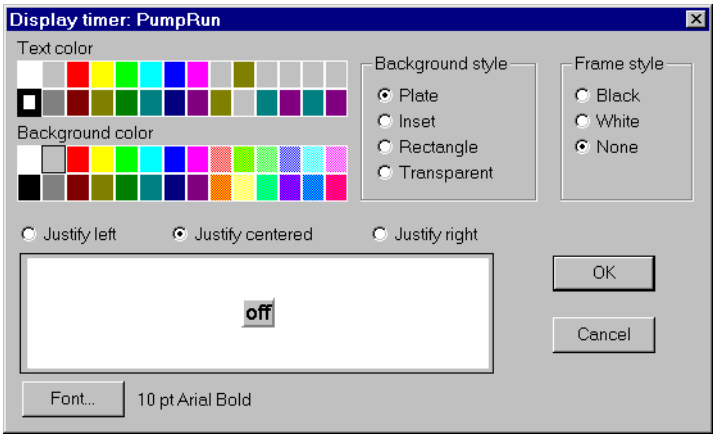


Figure 18-35. DelayOn Display Parameters Dialog Box

DelayOn Data Members

Table 18-37. DelayOn Data Members

Data Member	Type	Read	Write	Description
(implicit)	logical	yes	no	Logical timer value

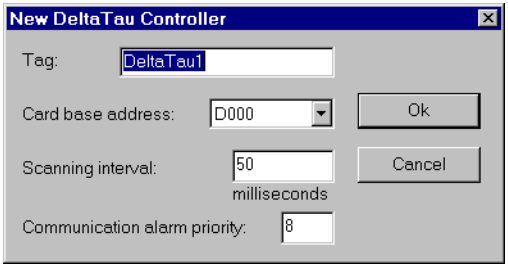
Comments ***The DelayOn timer can be used to prevent pumps from cycling too often, to allow one operation to complete before another begins, or to require a condition to exist for a period of time before an alarm is activated.***

Related Objects ***DelayOff, Interval, OneShot, Pulse, TimeOfxxx***

DeltaTau

DeltaTau is a protocol driver object class Lookout uses to communicate with Delta Tau Data Systems PMAC Motion Controllers. Create a DeltaTau object for each card installed in the computer.

This object class communicates with Delta Tau PMAC cards through dual-ported memory, so be sure that your PMAC hardware includes the dual-ported RAM option.



Delta Tau card configured to use PC memory beginning at address D000.

Card base address specifies the beginning memory location of the dual ported RAM address. It should match card settings.

Scanning Interval identifies the frequency that the DeltaTau object in Lookout polls the PMAC Motion Controller. Intervals can range from 10 ms to 1,000 ms.

Communication alarm priority specifies the priority level of alarms generated by the object.

DeltaTau Data members

Like other protocol driver objects, DeltaTau objects contain a great deal of data. All readable and writable members (inputs/outputs), polling instructions, and so on, are bundled with the object. Therefore, as soon as you create a DeltaTau object you immediately have access to all the object data members. The following table lists data members for the DeltaTau object

Table 18-38. DeltaTau Data Members

Data Member	Type	Read	Write	Description
DS0 – DS8190	numeric	yes	yes	Signed 32-bit word ranging from –2,147,483,648 to 2,147,483,647
DW0 – DW8190	numeric	yes	yes	32-Bit double-precision word ranging from 0 to 4,294,967,295
F0 – F8188	numeric	yes	yes	32-bit IEEE floating point word
S0 – S8190	numeric	yes	yes	Signed 16-bit word ranging from – 32,768 to 32,767
Update	logical	yes	no	Driver-generated signal that pulses each time Lookout scans the PMAC Motion Controller card
W0 – W8190	numeric	yes	yes	16-Bit word ranging from 0 to 65,535
W0.0 – W8190.15	logical	yes	yes	1 Bit in a 16-Bit word

Derivative

Derivative can also be called a rate of change object—it calculates the rate of change of the incoming numeric input signal. You can use this class to calculate the rate at which a tank is filling or draining, or to convert a changing totalized flow value into a flow rate. The output units are in Input Units/Time Unit.

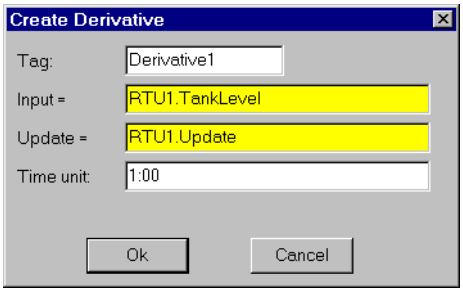


Figure 18-36. Derivative Definition Parameters Dialog Box

The above example calculates the rate of change in the water level of a tank. Lookout polls the RTU connected to the tank level transmitter every 5 minutes, so this example uses the RTU update data member as the Update pulse for the Derivative object. Tanklevel is in feet and Time unit is 1 minute, so the output result is in feet per minute.

Input is the numeric expression you are monitoring for rate of change.

Update can be a logical expression or numeric constant. If you specify **Update** as a numeric constant, it creates an internal pulse timer with a pulse period of the specified time and a pulse duration of zero. See *Numeric Data Members* in Chapter 5, *Developer Tour*, for information on entering time constants. If you specify **Update** as a logical variable, the variable should pulse at the frequency you want to use.

The **Update** expression triggers the calculation of a new rate-of-change based on the Input value at the prior Update, and the current Input value. The current Input value is then stored as the prior Input value for the next calculation. The Update period should be greater than the refresh rate of the incoming signal; or if the Input is generated directly by external I/O, the update data member generated by the PLC object should be used. If the Update period is less than the Input refresh rate, the rate of change calculation fluctuates erratically between zero and a high value.

Time unit is a numeric expression used as the basis for unit time on the Input signal. For instance, if the rate of change should be in feet per minute, the Input signal would be feet, and **Time unit** would be one minute (entered as 1:00). Typically the **Time unit** is one second (0:01), one minute (1:00), one hour (1:00:00), or one day (1:00:00:00). However, you can specify any unit, such as 5:23 (a rate of change in Input units per five minutes and 23 seconds).

☞ **Note** ***Derivative does not have a display parameters dialog box. You can display the result of a Derivative output signal by referencing its data member in an expression. When you complete the parameters in creating a Derivative object, an Insert Expression dialog box appears automatically.***

Derivative Data Members

Table 18-39. Derivative Data Members

Data Member	Type	Read	Write	Description
(implicit)	numeric	yes	no	Rate of change

Comments ***Derivative performs the inverse function of Integral—you can theoretically run a signal through an Integral object and then a Derivative object (or vice versa) and you would end up with the original signal. (Discretization of the time calculations by the computer may cause the final and original signals to differ somewhat).***

It is important to consider the resolution of the process variable measured by the PLC when determining the Update period for this object. For instance, if a pressure transmitter connected to a PLC only has a resolution of 0.5 psi and you want to measure rates down to 1 psi/minute, the Update pulse must be greater than 30 seconds even if the PLC is polled once per second (i.e., 0.5 psi/1 psi/min. = 30 sec.). For this application, the Update pulse should probably be about two minutes.

Related Objects ***Integral***

DialGauge

The DialGauge object class displays a numeric signal as a sweeping needle on an analog gauge or dial.

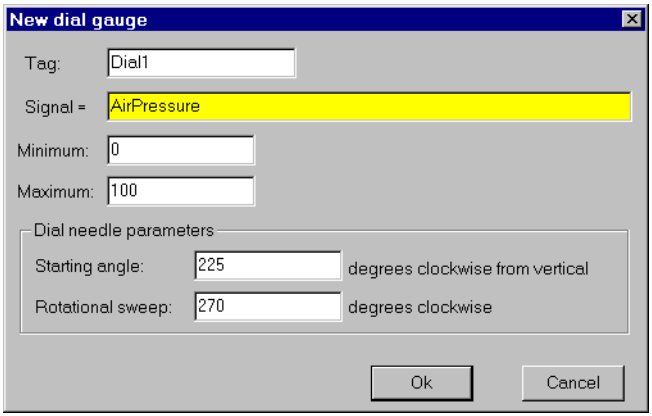
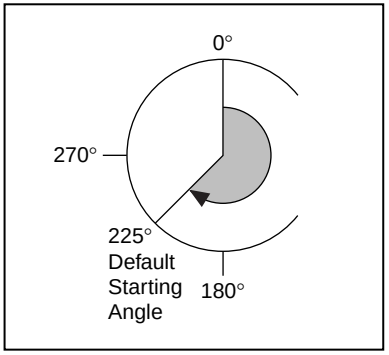


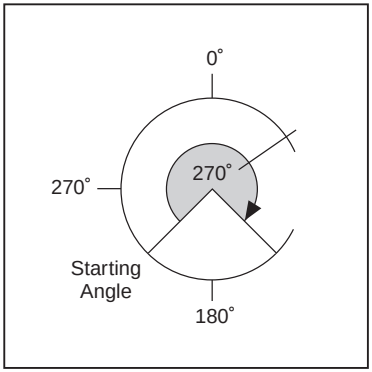
Figure 18-37. DialGauge Definition Parameters Dialog Box

Signal is a numeric expression.

Starting angle. indicates the position of the needle when the **Signal** is at its **Minimum** value. As shown here, you specify the starting needle position by counting the degrees clockwise from vertical.



Rotational sweep specifies the number of degrees clockwise that the needle will rotate as the **Signal** approaches the **Maximum** value. As shown in the diagram here, you count the **Rotational sweep** in degrees clockwise from the **Starting angle**.



After you specify the DialGauge definition parameters, Lookout presents a display parameters dialog box, as shown here. You use this dialog box to specify needle color, thickness, and length.

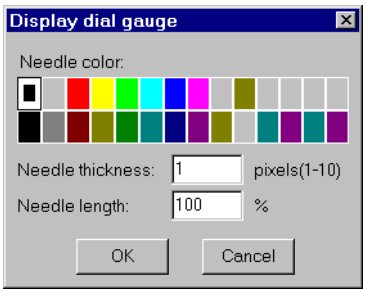
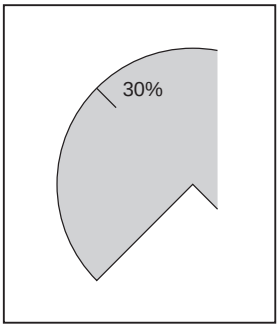


Figure 18-38. DialGauge Display Parameters Dialog Box

Needle thickness defines how wide your needle will be. Thickness can range from one pixel (hairline) to 10 pixels wide.

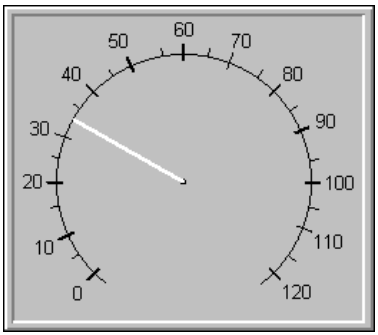
Needle length specifies the length of the needle as a percent of the radius. At 30 percent, for example, only the outer tip of the needle is visible—you cannot see the part of the needle closest to the origin. At 100 percent, the needle extends the full radius of the circle.



DialGauge Data Members

Table 18-40. DialGauge Data Members

Data Members	Type	Read	Write	Description
(implicit)	numeric	yes	no	Current value of signal parameter



Comments ***The DialGauge object class only displays a needle. You may wish to enhance it with a corresponding scale or dial face as shown. You can create a scale or dial face by importing one from a graphics package. See*** *Creating Custom Graphics* ***in Chapter 8,*** *Graphics****, for more information.***

☞ **Note** ***If you choose to import a scale or dial face from an external package, you should use a bitmap instead of a metafile.This makes the display refresh cleaner when the needle changes position.***

Related Objects ***Expression, Gauge***

DL205, DL405

DL205 and DL405 are protocol driver object classes Lookout uses to communicate with Koyo 205 and 405 PLCs, and other devices that use the CCM protocol.

Koyo sells the 205 and 405 models under private brand labels. PLCDirect offers the units as the DirectLOGIC 205 and 405 PLCs, and Siemens offers the 405 unit as the TI405 series.

The DL205 and DL405 object classes support the Koyos native protocol, CCM, which is also called DirectNET by PLCDirect and HostLink by Siemens. This documentation refers to this protocol as CCM.

The DL205 and DL405 object classes support both point-to-point and multidrop configurations. You can connect Lookout to a PLC programming port, built-in DirectNET/HostLink port, or to the PLC DCM (data communication module). These PLC communication ports use two different versions of the CCM protocol: K-sequence and N-sequence. Lookout supports both.

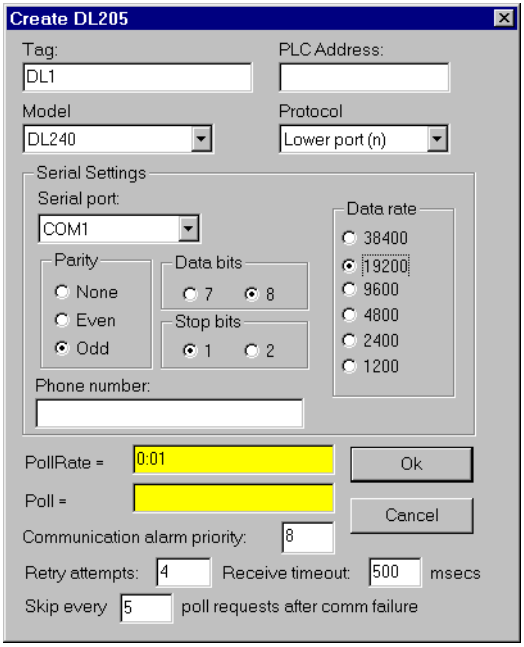


Figure 18-39. DL205 Parameters Configured for one PLC in a Multidropped Configuration

Protocol identifies the CCM port you use to communicate with the device. If you are planning to connect Lookout to a single PLC in a point-to-point configuration, choose the `UPPER PORT (k)` **Protocol** and use the PLC programming port. It supports K-sequence CCM. With this version of the CCM protocol you can write directly to logical outputs.

If you are planning to network multiple PLCs in a multidrop configuration where Lookout is the host computer, you must use the built-in PLC DirectNET/HostLink port, or its DCM. Choose the `LOWER PORT (n)` or `DCM (n)` *`Protocol`* as appropriate. These ports support N-sequence CCM. The N-sequence CCM protocol does not permit writes directly to logical data members.

☞ **Note** ***Although you can connect logical signals to data members such as Y0 in Lookout, any time you attempt to write to such a data member using the N-sequence CCM protocol, the write request is ignored. (This is due to the limitations of N-sequence CCM protocol.) If you have to write out a logical value using N-sequence, consider parsing the corresponding VU address into a binary number and setting the bit you want to use by writing a corresponding decimal integer. If you write a zero or 65535 to the corresponding VU-memory location, you will write all zeros or ones to every bit in the register.***

PLC Address is a slave address and refers to the address setting as set on the physical device. If you are connecting to the PLC programming port, set **PLC Address** to 1. If multiple devices share a common line, enter the unique addresses of the device (1 to 255).

Use **Model** to choose the PLC model you want to use. If you are using a model not listed, select a model whose data set most closely fits the model you are using.

Serial port specifies which COM port on your computer that the object uses for communicating to the PLC. This does not specify the communication type. Communication type is determined by the **Options»Serial Ports…** command.

Data rate, **Parity**, **Data bits**, and **Stop bits** should match the PLC settings.

Phone number specifies the number to be dialed if the selected serial port is configured for dial-up. This number only applies to the individual protocol object.

PollRate is a numeric expression that determines how often to poll the device. The object then polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *Introduction*, for information on entering time constants.

Poll is a logical expression. When this expression changes from false to true, Lookout polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Communication alarm priority determines the priority level of alarms generated by the object. Such alarms are typically related to communications with the physical device.

Retry attempts specifies the consecutive number of times Lookout attempts to establish communications with a device it does not get a valid response from. After the object retries communications the specified number of times, it generates a communication alarm and Lookout moves on to the next device in the polling queue (if any). Refer to Chapter 6, *Serial Communications*, for more information.

Receive timeout is the time delay Lookout uses in waiting for a response from a device before retrying the request.

The **Skip every…** setting instructs Lookout to not poll a device it has lost communication with on every scheduled poll. Instead, Lookout skips the device in the polling cycle accordingly. Once communications have been reestablished, the device is polled on its regular cycle.

DL205 and DL405 Data Members

Protocol driver objects contain a great deal of data. V-memory addresses, registers, inputs and outputs are all bundled with the object. Therefore, as soon as you create a DL205 object you have immediate access to the entire data member set of the object (see data member list below).

As with all Lookout drivers, you can access I/O points and other data through data members. The DL205 and DL405 object classes automatically generate an efficient read/write blocking scheme based on the inputs and outputs you are using in your process file. So, you do not have to build your own I/O blocking table. When Lookout polls a device, it optimizes frame exchange according to the data length required and overhead needed. The maximum data frame size that these drivers request is 256 data bytes. The drivers use LRC and parity error checking to validate data.

The following is a table of data members currently supported by the DL205 and DL405 object class.

Table 18-41. DL205 and DL405 Data Members

Data Member	Type	Read	Write	Description
C0 – C3777	logical	yes	yes	Control Relays—addressed in octal and mapped to V40600 – V40700
CommFail	logical	yes	no	Object-generated signal that is on if, for whatever reason, Lookout cannot communicate with the device
CT0 – CT377	logical	yes	yes	Counter status bits—addressed in octal and mapped to V41140 – V41147
OffHook	logical	no	yes	When TRUE, this flag instructs the DL object to retain exclusive use of its assigned communication port
Poll	logical	no	yes	When this value transitions from FALSE to TRUE, Lookout polls the device.

Table 18-41. DL205 and DL405 Data Members (Continued)

Data Member	Type	Read	Write	Description
PollRate	numeric	no	yes	Lookout expression that determines the device polling frequency.
S0 – S1777	logical	yes	yes	Stages—addressed in octal and mapped to V41000 – V41037
SPA0 – SPA177	logical	yes	yes	Lower special relays—addressed in octal
SPB320 – SPB717	logical	yes	yes	Upper special relays—addressed in octal
T0 – T377	logical	yes	yes	Timer status bits—addressed in octal and mapped to V41100 – V41107
Update	logical	yes	no	Object-generated signal that pulses low each time it polls the device
V0 – V41230	numeric	yes	yes	Signed V-memory register containing 16-bit integer ranging from –32768 to 32767 dec.
V0.0 – V41230.17	logical	yes	no	Individual bits within V-memory registers—the least significant bit is 0, the most significant bit is 17 (octal).
VBCD0-VBDC41230	numeric	yes	yes	V-memory as BCD
VD0 – VD41227	numeric	yes	yes	Double—reads two adjacent V-memory registers as a single 32-bit integer ranging from 0 to 4,294,967,296.
VDBCD0-VDBCD41227	numeric	yes	yes	V-memory double as BCD
VF0 – VF41227	numeric	yes	yes	Float—reads two adjacent V-memory registers as a single 32-bit floating point value.
VU0 – VU41230	numeric	yes	yes	Unsigned—V-memory register holding 16-bit unsigned integer ranging from 0 to 65535

Table 18-41. DL205 and DL405 Data Members (Continued)

Data Member	Type	Read	Write	Description
X0 – X1777*	logical	yes	yes	Discrete input points—addressed in octal and mapped to V40400 – V40407
Y0 – Y1777*	logical	yes	yes	Discrete output points—addressed in octal and mapped to V40500 – V40507
* 205 series PLCs are limited to 128 input and output points total. You can mix these addresses as necessary, but understand that Y17 and X17 refer to the same memory location on DL205.				

DL205 and DL405 Status Messages

No response within timeout period
Lookout did not receive the expected response within the **Receive timeout** period. The object sent an inquiry and received an acknowledgment, but the device did not send an expected response to the request. This might happen if the response was interrupted. You may have to increase **Receive timeout**.

No return inquiry response from secondary unit
Lookout received no response from the device within the **Receive timeout** period. The driver object is able to use the COM port, but when it polls the device, it does not respond—as if it is not even there. You may have to increase **Receive timeout** to ensure Lookout is allowing enough time to receive the expected response. Also, verify your baud rate settings, cable connections, power, configuration settings, COM port settings, and polling addresses.

Bad LRC
The object is receiving a poll response from the device, but it could not decipher the response because it is garbled. Verify that all devices connected to the COM port have unique addresses. The last part of the message may actually be getting clipped off before it is completed. Consider increasing the number of **Retry attempts**. You may have to increase the **Receive gap** Serial Port setting to ensure Lookout is receiving the entire message. If your Serial Port is configured for radio, this could be caused by an audible squelch tail occurring at the end of a radio transmission. Try adjusting **RTS delay off** and **CTS timeout**.

No acknowledgment for header frame
You sent an inquiry and some kind of data write or read, but the device has not responded. The object was expecting an acknowledgment for an instruction frame and did not receive it. You may be asking for an invalid memory address. Recheck your PLC memory address configuration. Also verify that the PLC communication port supports the CCM **Protocol** that you selected.

Invalid request frame
You are probably trying to use an invalid memory address. Recheck your PLC address configuration.

Dynamic

Dynamic is a protocol driver class Lookout uses to communicate with equipment such as programmable logic controllers (PLCs), remote terminal units (RTUs), or any other piece of equipment that uses Dynamic as its communication protocol.

Protocol driver objects contain a great deal of data. All readable and writable members (inputs/outputs), polling instructions, read/write blocking, serial port usage, etc. are bundled with the object. Therefore, as soon as you create a Dynamic object you immediately have access to all the object data members (see data member list below).

The driver currently supports reading and writing of all predefined data point types including analog input, state input, pulse count input, pulse duration input, pulse period input, high speed accumulator inputs, state outputs, pulse outputs, analog outputs, and AGA data.

The driver automatically configures the RTU database to match the data points referenced in the Lookout process file and verifies that the points specified in the process file actually exist on the RTU. If you reference undefined or unavailable points, the driver posts an alarm message to the Lookout alarm system.

The Dynamic protocol driver automatically takes advantage of the RTU exception reporting capabilities to maximize communication efficiency. The driver requires that the RTU be set for physical point addressing—refer to the RTU hardware technical reference manual for the dipswitch locations that set this parameter.

☞ **Note** ***Lookout protocol driver objects automatically generate an efficient read/write blocking scheme based on the inputs and outputs being used in your process file. You are not required to build your own I/O blocking table.***

The Dynamic protocol driver was written for Texas Instruments Model 8635, 8640 and 8641 RTUs (Remote Terminal Units) with RTX 5.41 EPROMs—earlier releases of RTX have significant bugs and do not operate correctly with the Lookout Dynamic driver.

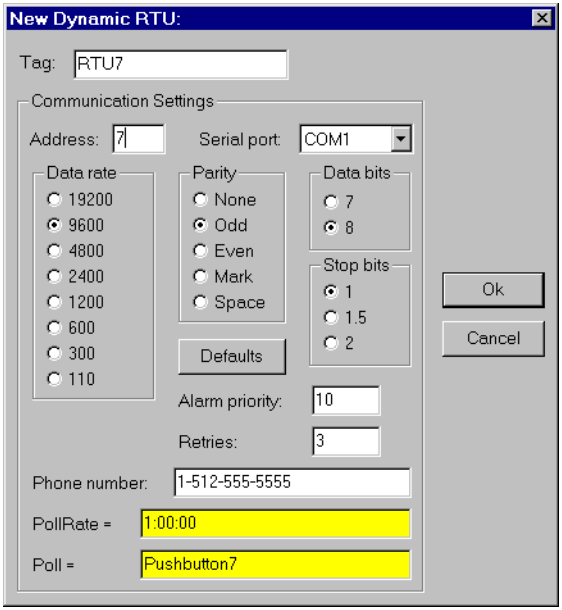


Figure 18-40. Dynamic Parameter Configuration Dialog Box

In the example shown above, Lookout is connected to a Dynamic-speaking RTU with an address of 7. Lookout is using serial port1 (which was previously configured for Dial-up communications), and calling the specified phone number. Each poll occurs every 1 hour or when the operator depresses Pushbutton7.

Address refers to the PLC or RTU address setting as set on the device dipswitches. If devices share a common line, they require unique addresses (1 to 255).

Serial port specifies which port the object uses for communication to the external device. This does not specify the communication type. Communication type is determined by the **Options»Serial Ports…** command.

Data rate, **Parity**, **Data bits**, and **Stop bits** reference the settings on the hardware device.

The **Defaults** button replaces the current settings with default values.

Alarm priority determines the priority level of Dynamic-generated alarms.

Phone number specifies the number to be dialed if the serial port setting is configured for dial-up. This number only applies to the individual protocol object.

PollRate is a numeric expression that determines how often to poll the device. Dynamic then polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *Introduction*, for information on entering time constants.

Poll is a logical expression. When this expression changes from false to true, Lookout polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

As with all Lookout drivers, you can access I/O points and other data through data members. The following is a table of data members currently supported by the Dynamic object class.

Dynamic Data Members

Table 18-42. Dynamic Data Members

Data Members	Type	Read	Write	Description
AI0 – AI239	numeric	yes	no	Analog inputs—returns a normalized numeric value that ranges from 0 to 32000 and spans the full input analog range.
AO0 – AO239	numeric	no	yes	Analog outputs—sets voltage level of analog outputs, and accepts a normalized number from 0 to 32000.
btu1 – btu4	numeric	yes	no	AGA data
dp1 – dp4	numeric	yes	no	Differential pressure (psi). 1 – 4 specifies meter run being referenced.
HSA0 – HSA127	numeric	yes	no	High-speed accumulator inputs—counts the number of digital pulses on an HSA input. Rollover is detected automatically by driver, so maximum pulse count is 4.5×10^{15}.

Table 18-42. Dynamic Data Members (Continued)

Data Members	Type	Read	Write	Description
MeterCode1 – MeterCode4	text	no	yes	First 8 characters of this text string are used to create a filename with a .CSV extension. Entire text string is used as a field within the .CSV file. 1 – 4 specifies meter run being referenced.
NoComm	logical	yes	no	Driver-generated signal that is on if Lookout cannot communicate with the device for whatever reason
PCI0 – PCI239	numeric	yes	no	Digital pulse count inputs—counts the number of times a digital state input has transitioned from `off` to `on`. Rollover is detected automatically by driver, so maximum pulse count is about 4.5×10^{15}.
PDI0 – PDI239	numeric	yes	no	Digital pulse duration inputs—measures a periodic pulse on time with a 10 msec resolution. Pulse period should be 15 seconds or less. Returned time is in standard Lookout time units of fraction of a day.
Poll	logical	no	yes	When this value transitions from FALSE to TRUE Lookout polls the device.
PollRate	numeric	no	yes	Lookout expression that determines the frequency at which the device is to be polled.
PPI0 – PPI239	numeric	yes	no	Digital pulse period inputs—measures a pulse period with a 10 msec resolution. Pulse period should be 15.1 seconds or less. Returned time is in standard Lookout time units of fraction of a day.
ReadHistory	logical	no	yes	When this value transitions from TRUE to FALSE, Lookout polls the device and reads the historical AGA from the RTU, storing to disk.

Table 18-42. Dynamic Data Members (Continued)

Data Members	Type	Read	Write	Description
Qe1 – Qe4	numeric	yes	no	Energy rate 1 – 4 specifies meter run being referenced.
Qsum_month1 – Qsum_month4	numeric	yes	no	Accumulated flow over the last 30 days (cubic feet – CF). 1 – 4 specifies meter run being referenced.
Qsum_yesterday1– Qsum_yesterday4	numeric	yes	no	Accumulated flow over the last 24 hours (cubic feet – CF). 1 – 4 specifies meter run being referenced.
Qsum1 – Qsum4	numeric	yes	no	Accumulated flow over the last 60 minutes (cubic feet – CF). 1 – 4 specifies meter run being referenced.
Qv1 – Qv4	numeric	yes	no	Instantaneous flow rate (cubic feet/hour – CFH). 1 – 4 specifies meter run being referenced.
sp1 – sp4	numeric	yes	no	Static pressure (psi). 1 – 4 specifies meter run being referenced.
tf1 – tf4	numeric	yes	no	Flowing temperature (deg. F). 1 – 4 specifies meter run being referenced.
SI0 – SI239	logical	yes	no	Digital state inputs
SO0 – SO239	logical	no	yes	Digital state outputs. On the 8635, state outputs and pulse outputs are mapped to programmable I/O (PIO) points and range from SO8 to SO15.
Update	logical	yes	no	Driver-generated signal that pulses each time the driver polls the device
VB0-VB65535	logical	yes	yes	Bit in V register read as logical.
VF0-VF2047	numeric	yes	yes	V register read as float on 4-byte boundary.
VFN0-VFN2047	numeric	yes	yes	V register read as float, no boundary.

Table 18-42. Dynamic Data Members (Continued)

Data Members	Type	Read	Write	Description
VW0-VW4095	numeric	yes	yes	V register read as a word on 2-byte boundary.
VWN0-VWN4095	numeric	yes	yes	V register read as a word, no boundary.

ElapsedTime

ElapsedTime is an elapsed time meter or "hour meter" that totals the amount of time the **Enable** expression is on. If **Enable** is the logical constant ON, the meter reflects the time since the process was started. If a **Reset** expression is specified, the meter resets to zero the moment **Reset** transitions from off to on. The display always shows the elapsed time, and is updated approximately once per second.

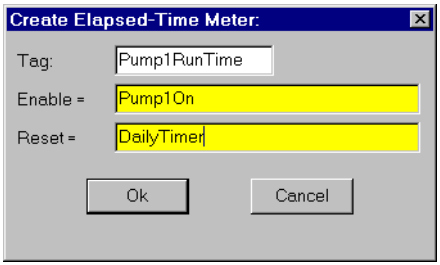


Figure 18-41. ElapsedTime Definition Parameters Dialog Box

Note ***ElapsedTime does not have a display parameters dialog box. However, you can easily display the result of the ElapsedTime output signal by referencing its data member in an expression.***

ElapsedTime Data Members

Table 18-43. ElapsedTime Data Members

Data Member	Type	Read	Write	Description
(implicit)	numeric	yes	no	Total elapsed time—updated once per second, while meter is running

Comments ***ElapsedTime meters are used primarily to record the amount of time that individual pieces of equipment have been running. It is also straightforward to set up an alarm that sounds when a particular device has been operating for a certain time and needs routine servicing. The plant operator could then perform the service and reset the ElapsedTime meter with a pushbutton.***

ElapsedTime can also record the amount of time a particular device is operated each day and you can record the resulting time to a daily Spreadsheet which you can then use to automatically reset the meter after the data is permanently recorded.

Event

Event is a flexible and powerful object class you can use to define event messages that are triggered based on a user-defined logical expression. Lookout logs such events to the `event.dat` log file and you can subsequently print and archive them. See Chapter 11, *Logging Data and Events*, for more information on logging events.

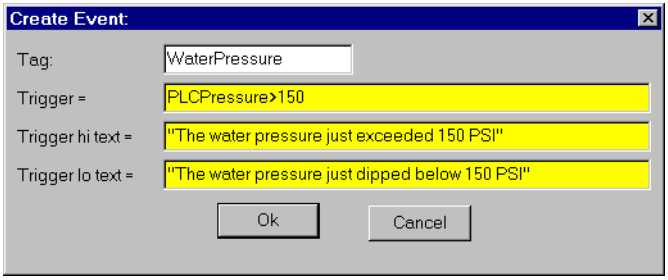


Figure 18-42. Typical Settings for an Event

When the result of the **Trigger** logical expression transitions from FALSE to TRUE, it logs the result of the **Trigger hi text** expression as an event in the `EVENT.DAT` file. When the results of the **Trigger** expression transitions from TRUE to FALSE, it logs the results of the **Trigger lo text** expression as an event.

Event Data Members

Table 18-44. Event Data Members

Data Member	Type	Read	Write	Description
none	—	—	—	Event objects do not have data members

Comments ***For each event logged, Lookout records the date and time, the name of the user currently logged on, and the expression text.***

Although event messages are shown for both **Trigger hi text** and **Trigger lo text**, you do not have to include text in both fields.

Expression

Expressions are flexible, powerful real-time calculators. They create and calculate the result of spreadsheet-style formulas that include a mixture of constants and signals from other objects. There are over fifty built-in functions that you use in expressions, including logical, mathematical, statistical, text and trigonometric functions. See Chapter 7, *Expressions*, for more information on expressions and expression functions.

Expressions can be short and simple, or extremely complex with several signal inputs, function calls, and multiple levels of parentheses. A single expression may incorporate text, logical, and numeric calculations. The variable type returned by the outermost function or operator in the expression determines the signal type generated by the expression.

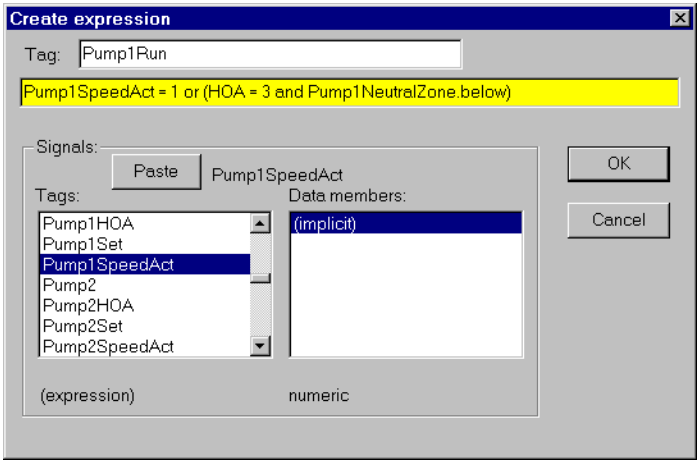


Figure 18-43. Create Expression Dialog Box

Note ***You typically use Expression objects when you need to define a unique condition that is used multiple times throughout your application.***

When you define an Expression object (as opposed to inserting an intrinsic expression), you create a unique tagname for your expression and can therefore reference the output signal generated from the expression in other expressions or objects. Instead of defining the same expression in many places, you create it one time and use its tagname any time you need this expression.

Note ***The expression may not express a* condition.**

☞ **Note** ***An Expression object does not have a display parameters dialog box. You can easily display the Expression object output signal by referencing its data member in an intrinsic expression.***

Table 18-45. Expression Data Members

Data Member	Type	Read	Write	Description
(implicit)	numeric, logical, or text	yes	no	Value of expression. The variable type returned by the outermost function or operator in the expression determines the signal type generated by the expression.

FisherROC

FisherROC is a protocol driver class Lookout uses to communicate with a ROC364 (Remote Operations Controller) made by Fisher Controls. To communicate with the ROC364, connect directly to its COM1 or COM2 port. Depending on the device configuration, these ports may be configured for RS-232, RS-422/485 or Bell 202 modem communications.

Create one FisherROC object for each ROC364. This object class contains a great deal of data. All point types including inputs, outputs, PID, AGA, Tank, and their configuration signals are bundled with the object. Therefore, as soon as you create a FisherROC object you immediately have access to the object data member set (see data member list below).

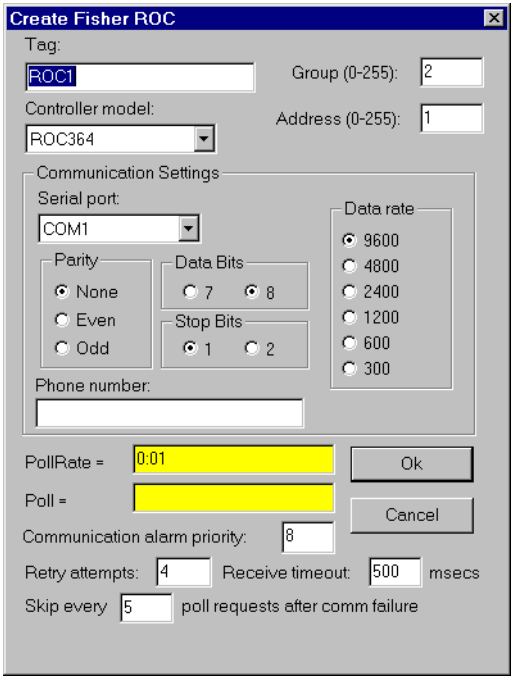


Figure 18-44. FisherROC Definition Parameters Dialog Box

Group is the group code to which the ROC station is assigned. This is typically set to 2, but is configurable through the GV101 Configuration software.

Address is the unit code (address setting) of the ROC station as defined using the GV101 Configuration software. If multiple ROCs are members

of the same group, then they must have unique addresses (0 to 255). As the host station, the Lookout computer is assigned Group 0, Address 1.

Controller model identifies the type of Fisher remote operations controller that the object represents. This object class supports **ROC364**.

Serial port specifies which COM port on the host computer that Lookout object uses for communicating to the external device. This does not specify the communication type. Communication type is determined by the **Options»Serial Ports…** command.

Data rate, **Parity**, **Data bits**, and **Stop bits** reference the settings on the hardware device.

Phone number specifies the number to be dialed if the selected serial port is configured for dial-up. This number only applies to the individual protocol object.

PollRate is a numeric expression that determines how often to poll the device. The object then polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *Introduction*, for information on entering time constants.

Poll is a logical expression. When this expression changes from FALSE to TRUE, Lookout polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Communication alarm priority determines the priority level of alarms generated by the object. Such alarms are typically related to communications with the physical device.

Retry attempts specifies the consecutive number of times Lookout attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the object generates an alarm and releases the communication port back to the communications service which then moves on to the next device in the polling queue (if any). Refer to Chapter 6, *Serial Communications*, for more information.

Receive timeout is the time delay Lookout uses in waiting for a response from a device before retrying the request.

The **Skip every…** setting instructs Lookout to not poll a device it has lost communication with on every scheduled poll. Instead, Lookout skips the

device in the polling cycle accordingly. Once communications have been reestablished, the device is polled on its regular cycle.

FisherROC Data Members

Table 18-46. FisherROC Data Members

Data Member	Type	Read	Write	Description
AGA1.la – AGA5.la	numeric	yes	yes	AGA low alarm EU value
AGF1.btud – AGF5.btud	numeric	yes	no	AGA Flow in units of MMBTU/Day
AGF1.btut – AGF5.btut	numeric	yes	no	MMBTUs of AGA Flow today
AGF1.btuy – AGF5.btuy	numeric	yes	no	MMBTUs of AGA Flow yesterday
AGF1.mcfd – AGF5.mcfd	numeric	yes	no	AGA Flow in units of MCF/Day
AGF1.mcft – AGF5.mcft	numeric	yes	no	MCFs of AGA Flow today
AGF1.mcfy – AGF5.mcfy	numeric	yes	no	MCFs of AGA Flow yesterday
AI:A1.adh – AI:D16.adh	numeric	yes	yes	Analog input adjusted A/D 100%
AI:A1.adl – AI:D16.adl	numeric	yes	yes	Analog input adjusted A/D 0%
AI:A1.dla – AI:D16.dla	numeric	yes	yes	Analog input Delta Alarm EU
AI:A1.euh – AI:D16.euh	numeric	yes	yes	Analog input High Reading EU
AI:A1.eul – AI:D16.eul	numeric	yes	yes	Analog input Low Reading EU
AI:A1.feu – AI:D16.feu	numeric	yes	yes	Analog input Filtered EU value
AI:A1.flt – AI:D16.flt	numeric	yes	yes	Analog input Filter
AI:A1.ha – AI:D16.ha	numeric	yes	yes	Analog input High alarm limit
AI:A1.hha – AI:D16.hha	numeric	yes	yes	Analog input High-High alarm limit
AI:A1.la – AI:D16.la	numeric	yes	yes	Analog input Low alarm limit
AI:A1.lla – AI:D16.lla	numeric	yes	yes	Analog input Low-Low alarm limit
AI:A1.raw – AI:D16.raw	numeric	yes	no	Analog input Raw A/D input value
AO:A1.adh – AO:D16.adh	numeric	yes	yes	Analog output Adjusted A/D 100%
AO:A1.adl – AO:D16.adl	numeric	yes	yes	Analog output Adjusted A/D 0%
AO:A1.eu – AO:D16.eu	numeric	yes	yes	Analog output EU output value

Table 18-46. FisherROC Data Members (Continued)

Data Member	Type	Read	Write	Description
AO:A1.euh – AO:D16.euh	numeric	yes	yes	Analog output High Reading EU
AO:A1.eul – AO:D16.eul	numeric	yes	yes	Analog output Low Reading EU
CommFail	logical	yes	no	Object-generated signal that is on if, for whatever reason, Lookout cannot communicate with the ROC
DI:A1.acc – DI:D16.acc	numeric	yes	yes	Discrete input accumulated value
DI:A1.flt – DI:D16.flt	logical	yes	yes	Discrete input filter
DI:A1.ha – DI:D16.ha	numeric	yes	yes	Discrete input high alarm
DI:A1.hha – DI:D16.hha	numeric	yes	yes	Discrete input high-high alarm
DI:A1.la – DI:D16.la	numeric	yes	yes	Discrete input low alarm
DI:A1.lla – DI:D16.lla	numeric	yes	yes	Discrete input low-low alarm
DI:A1.sts – DI:D16.sts	logical	yes	no	Discrete input status (value)
DO:A1.acc – DO:D16.acc	numeric	yes	yes	Discrete output accumulated value
DO:A1.cth – DO:D16.cth	numeric	yes	yes	Discrete output 100% count
DO:A1.ctl – DO:D16.ctl	numeric	yes	yes	Discrete output 0% count
DO:A1.ctm – DO:D16.ctm	numeric	yes	yes	Discrete output cycle time
DO:A1.eu – DO:D16.eu	numeric	yes	yes	Discrete output EU value
DO:A1.rdh – DO:D16.rdh	numeric	yes	yes	Discrete output high reading EU
DO:A1.rdl – DO:D16.rdl	numeric	yes	yes	Discrete output low reading EU
DO:A1.sts – DO:D16.sts	logical	yes	yes	Discrete output status (value)
DO:A1.to – DO:D16.to	numeric	yes	yes	Discrete output time on
FST1.rg1 – FST8.rg10	numeric	yes	yes	Function seq. table register value
FST1.rrg – FST8.rrg	numeric	yes	yes	Function seq. table result register
FST1.tm1 – FST8.tm4	numeric	yes	yes	Function seq. table timer value
PID1.eu – PID16.eu	numeric	yes	yes	PID output EU value
PID1.ost – PID16.ost	numeric	yes	yes	PID OVR sw setpoint

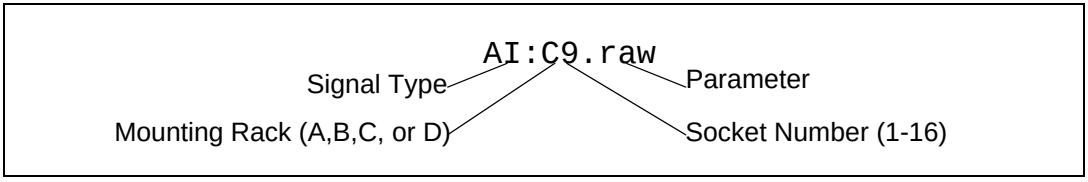
Table 18-46. FisherROC Data Members (Continued)

Data Member	Type	Read	Write	Description
PID1.pst – PID16.pst	numeric	yes	yes	PID PRI sw setpoint
PID1.set – PID16.set	numeric	yes	yes	PID setpoint value
PL:A1.acc – PL:D16.acc	numeric	yes	yes	Pulse input accumulated value
PL:A1.crt – PL:D16.crt	numeric	yes	no	Pulse input current rate
PL:A1.dla – PL:D16.dla	numeric	yes	yes	Pulse input delta alarm EU
PL:A1.eu – PL:D16.eu	numeric	yes	yes	Pulse input value in engineering units
PL:A1.ha – PL:D16.ha	numeric	yes	yes	Pulse input high alarm EU value
PL:A1.hha – PL:D16.hha	numeric	yes	yes	Pulse input high-high alarm EU value
PL:A1.la – PL:D16.la	numeric	yes	yes	Pulse input low alarm EU value
PL:A1.lla – PL:D16.lla	numeric	yes	yes	Pulse input low-low alarm EU value
PL:A1.rpd – PL:D16.rpd	numeric	yes	yes	Pulse input rate period
Poll	logical	no	yes	When this value transitions from FALSE to TRUE, Lookout polls the device.
PollRate	numeric	no	yes	Lookout expression that determines the device polling frequency.
SPT1.d1 – SPT32.d20	numeric	yes	yes	Soft point parameter data number
TNK1.cfl – TNK8.cfl	numeric	yes	no	Tank current fluid level
TNK1.dis – TNK8.dis	numeric	yes	no	Tank barrels discharged
TNK1.dla – TNK8.dla	numeric	yes	yes	Tank delta alarm EU
TNK1.lsl – TNK8.lsl	numeric	yes	no	Tank last scan level
TNK1.man – TNK8.man	numeric	yes	yes	Tank manual entry—barrels
TNK1.ttl – TNK8.ttl	numeric	yes	no	Tank total barrels hauled
TNK1.tvl – TNK8.tvl	numeric	yes	no	Today’s tank volume

Table 18-46. FisherROC Data Members (Continued)

Data Member	Type	Read	Write	Description
TNK1.yvl – TNK8.yvl	numeric	yes	no	Yesterday's tank volume
Update	logical	yes	no	Object-generated signal that pulses low each time it polls the device

Comments ***To read the raw value of an analog input located at I/O module socket 9 on the third mounting rack, you would enter*** `TAGNAME.AI:C9.raw`***, where***



FisherROC Status Messages

No response within timeout period
Lookout received no response from the ROC device within the **Receive timeout** period. The driver object is able to use the COM port, but when it polls the device, it does not respond—as if it is not even there. You may have to increase **Receive timeout** to ensure Lookout is allowing enough time to receive the expected response. Also, verify your baud rate settings, cable connections, power, configuration settings, COM port settings, and polling addresses.

Message Garbled—Bad CRC
The object is receiving a poll response from the device, but it cannot decipher the response. Verify that all devices assigned to the device group have unique unit codes. The last part of the message may actually be getting clipped off before it is completed. Consider increasing the number of **Retry attempts**. You may have to increase the **Receive gap** Serial Port setting to ensure Lookout is receiving the entire message. If your Serial Port is configured for radio, this could be caused by an audible squelch tail occurring at the end of a radio transmission. Try adjusting **RTS delay off** and **CTS timeout**.

Unexpected data response length
The object is receiving a poll response from the device, but the response is too long. Verify that all devices connected to the COM port and assigned to the same device group have unique unit codes. Devices with identical

addresses may be trying to respond at the same time. Also verify device protocol settings.

Response too short
The object is receiving data of a length that does not meet the minimum frame length requirements. Verify device protocol settings. You may have to increase **Receive timeout**.

Return unit address incorrect
Return group address incorrect
The object sent a data write or data read request, but the ROC did not respond. You may be asking for an invalid memory address. Recheck your ROC address configuration.

Invalid module or point requested
You attempted a data type mismatch. For example, you may have tried to write or read an *analog* value to or from a rack I/O socket that has a *discrete* module installed, or visa versa. This is an addressing problem. Verify that the type of I/O at the desired socket matches the signal type of the address identified in the data member you are writing to.

Unexpected opcode in response: xx
The object is receiving a poll response from the device, but the response was not expected. This might happen if the object receives a poll response after the **Receive timeout** period expires. You may have to increase **Receive timeout**. Another possibility is that the ROC is attempting an unsolicited communication. The FisherROC object class currently does not support unsolicited report-by-exception.

Flipflop

Flipflop changes its logical output signal from on to off, or from off to on when the **Input** signal goes high. The output signal does not change when the signal goes low. **Input** is a logical expression.

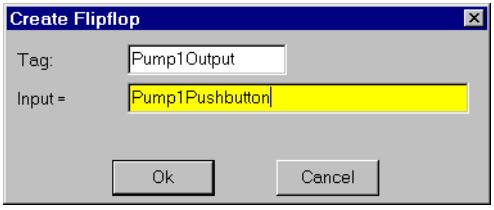


Figure 18-45. Flipflop Definition Parameters Dialog Box

FlipFlop Data Members

Table 18-47. Flipflop Data Members

Data Member	Type	Read	Write	Description
(implicit)	logical	yes	no	Current state

Comments ***Flipflop can be used to alternate the operation of two pumps, or when connected to a pushbutton, provides a pushbutton on/off control device.***

Related Objects ***LatchGate***

Gauge

Gauge displays the **Signal** expression in digital or bar graph format. Gauge display parameters change depending on the values of the **Conditional expressions**. Gauge determines which colors to display based on the order and current status of your conditional expressions. For instance, if several conditional expressions are true at once, Gauge displays the color associated with the first true expression.

You can use the **Transparent** background style with numeric expressions and gauges displayed as bar graphs. This means you can have bar graphs with transparent backgrounds.

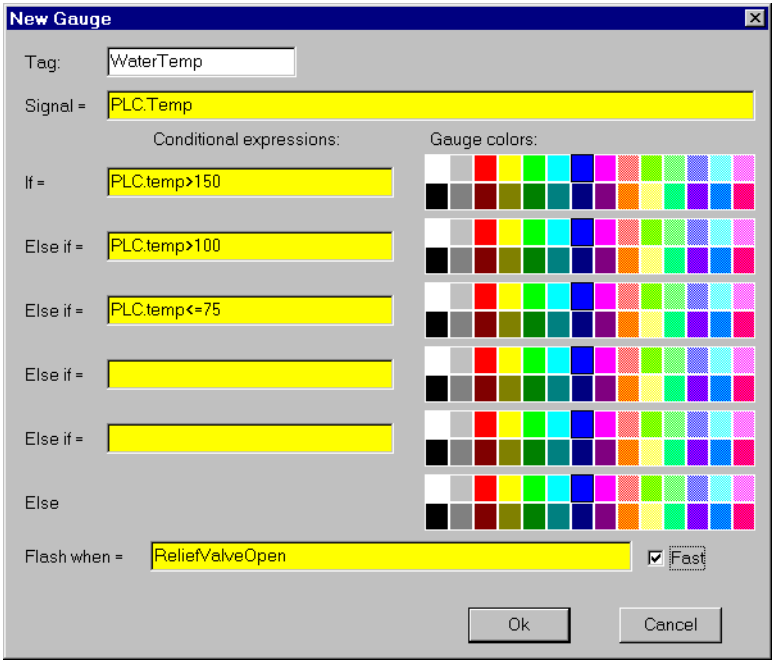


Figure 18-46. Gauge Definition Parameters Dialog Box

Conditional expressions and **Flash when** are logical expressions while **Signal** is a numeric expression. The **Fast** option instructs the Gauge to flash faster when enabled than when disabled.

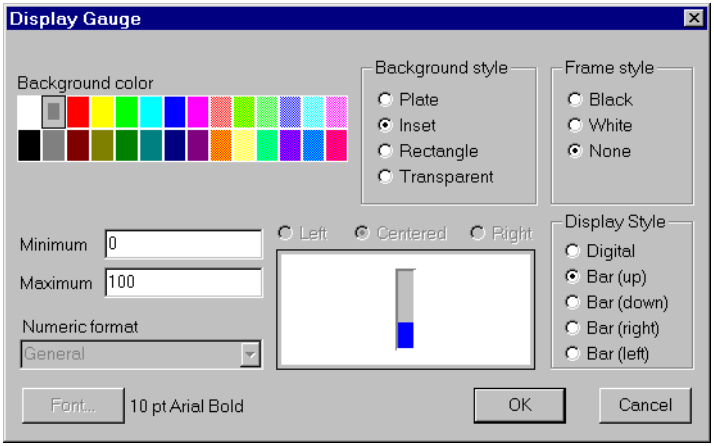


Figure 18-47. Gauge Display Parameters Dialog Box

Gauge Data Members

Table 18-48. Gauge Data Members

Data Member	Type	Read	Write	Description
(implicit)	numeric	yes	no	Numeric value of Gauge

Comments ***You should use a Gauge object when you need a bar graph or digital display to change colors and/or flash upon certain conditions. If you do not need either of these capabilities, you should display a bar graph or digital value with the* Insert»Expression… *command.***

GE_Series6

GE_Series6 is a protocol driver class Lookout uses to communicate with GE Series Six PLCs and other equipment using the CCM communication protocol. This driver handles both point-to-point and multidrop configurations in CCM mode, and can be connected to CCM2 or CCM3 communication control modules via either RS-232 or RS-422 serial communication ports.

☞ **Note** ***Use a Modbus object if you want to communicate with a GE Series Six PLC in RTU mode.***

The GE_Series6 object class automatically generates an efficient read/write blocking scheme based on the inputs and outputs being used in your process file. You are not required to build your own I/O blocking table. When Lookout polls a device, it optimizes frame exchange according to the data length required and overhead needed. The maximum data frame size that this driver requests is 256 data bytes. It uses LRC and parity error checking to validate data.

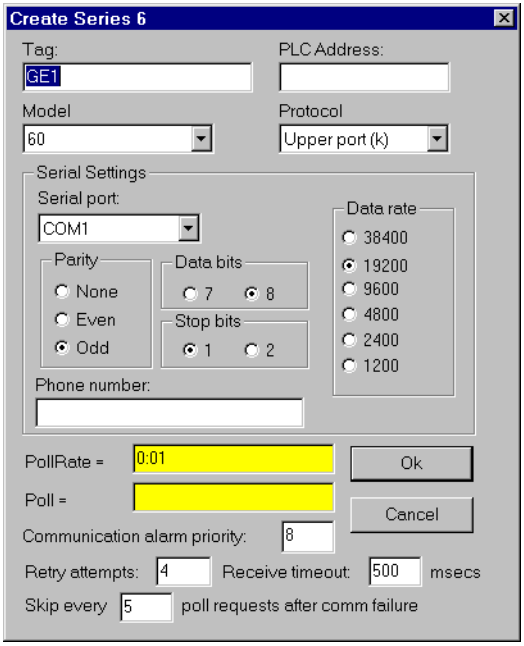


Figure 18-48. GE_Series6 Definition Parameters Dialog Box

PLC Address is a slave address and refers to the address setting as set on the device dip switches. If devices share a common line, they require unique addresses (1 to 255).

Model chooses from among the following PLC models: `60`, `600`, `6000`, `Plus`, and `Plus II`. If you are using a model not listed, select a model whose data set most closely fits the model you are using.

Protocol identifies the communication protocol that you use to communicate with the device. `CCM` is currently the only one available for this object class.

Serial port specifies which COM port the object uses for communicating to the external device. This does not specify the communication type. Communication type is determined by the **Options»Serial Ports…** command.

Data rate, **Parity**, **Data bits**, and **Stop bits** reference the settings on the hardware device.

Phone number specifies the number to be dialed if the selected serial port is configured for dial-up. This number only applies to the individual protocol object.

PollRate is a numeric expression that determines how often to poll the device. The object then polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *Introduction*, for information on entering time constants.

Poll is a logical expression. When this expression changes from FALSE to TRUE, Lookout polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Communication alarm priority determines the priority level of alarms generated by the object. Such alarms are typically related to communications with the physical device.

Retry attempts specifies the consecutive number of times Lookout attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the object generates an alarm and releases the communication port back to the communications service which then moves on to the next device in the polling queue (if any). Refer to Chapter 6, *Serial Communications*, for more information.

Receive timeout is the time delay Lookout uses in waiting for a response from a device before retrying the request.

The **Skip every…** setting instructs Lookout to not poll a device it has lost communication with on every scheduled poll. Instead, Lookout skips the device in the polling cycle accordingly. Once communications have been reestablished, the device is polled on its regular cycle.

This protocol driver object contains a great deal of data. The register table and input and output tables are bundled with the object. As soon as you create a GE_Series6 object you immediately have access to the object data member set (see data member list below).

GE_Series6 Data Members

As with all Lookout drivers, you can access I/O points and other data through data members. The following is a table of data members currently supported by the GE Series Six object class.

Table 18-49. GE_Series6 Data Members

Data Member	Type	Read	Write	Description
CommFail	logical	yes	no	Object-generated signal that is on if, for whatever reason, Lookout cannot communicate with the device
I0 – I128	logical	yes	no	Discrete inputs
O0–O128	logical	yes	yes	Discrete outputs
OffHook	logical	no	yes	When TRUE, this flag instructs the GE object to retain exclusive use of its assigned communication port
Poll	logical	no	yes	When this value transitions from FALSE to TRUE, Lookout polls the device.
PollRate	numeric	no	yes	Lookout expression that determines the device polling frequency.
R0 – R16384	numeric	yes	yes	Signed. 16-bit registers encoded as signed integers ranging from –32767 to +32768

Table 18-49. GE_Series6 Data Members (Continued)

Data Member	Type	Read	Write	Description
R0.0 – R16384.15	logical	yes	no	Individual bits within registers—reads logical on/off values. The least significant bit is 1; the most significant, 15.
RD0 – RD16383	numeric	yes	yes	Double. 32-bit unsigned register—reads two adjacent registers as a single 32-bit number ranging from 0 to 4,294,967,296.
RF0 – RF16383	numeric	yes	yes	Float. 32-bit IEEE floating point register—reads two adjacent registers as a single 32-bit floating point value
RU0 – RU16384	numeric	yes	yes	Unsigned. 16-bit registers holding unsigned integers ranging from 0 to 65535
Update	logical	yes	no	Object-generated signal that pulses low each time it polls the device

GE_Series6 Status Messages

No response within timeout period
Lookout did not received the expected response within the **Receive timeout** period. The object sent an inquiry and received an acknowledgment, but the device did not send an expected response to the request. This might happen if the response was interrupted. You may have to increase **Receive timeout**.

No return inquiry response from secondary unit
Lookout received no response from the device within the **Receive timeout** period. The driver object is able to use the COM port, but when it polls the device, it does not respond—as if it is not even there. You may have to increase **Receive timeout** to ensure Lookout is allowing enough time to receive the expected response. Also, verify your baud rate settings, cable connections, power, configuration settings, COM port settings, and polling addresses.

Bad LRC
The object is receiving a poll response from the device, but it could not decipher the response because it is garbled. Verify that all devices connected to the COM port have unique addresses. The last part of the

message may actually be getting clipped off before it is completed. Consider increasing the number of **Retry attempts**. You may have to increase the **Receive gap** Serial Port setting to ensure Lookout is receiving the entire message. If your Serial Port is configured for radio, this could be caused by an audible squelch tail occurring at the end of a radio transmission. Try adjusting **RTS delay off** and **CTS timeout**.

No acknowledgment from header frame
You sent an inquiry and some kind of data write or read, but the device has not responded. The object was expecting an acknowledgment for an instruction frame and did not receive it. You may be asking for an invalid memory address. Check your PLC address configuration. This object class speaks N Sequence CCM. If you are connected to the PLC port normally used for programming, you may be trying to speak to a port configured for K Sequence CCM. If another port is available, try plugging into that one. Otherwise, reconfigure the PLC port for N Sequence CCM.

Invalid request frame
You are probably trying to use an invalid memory address. Check your PLC address configuration.

GE_Series90

GE_Series90 is a protocol driver class Lookout uses to communicate with GE Series 90-30 and GE Series 90-70 programmable logic controllers (PLCs) using SNPX, a Series Ninety Protocol.

☞ **Note** ***This object class is available on Version 3.6 (build 10) and later. It is not backward compatible to earlier versions.***

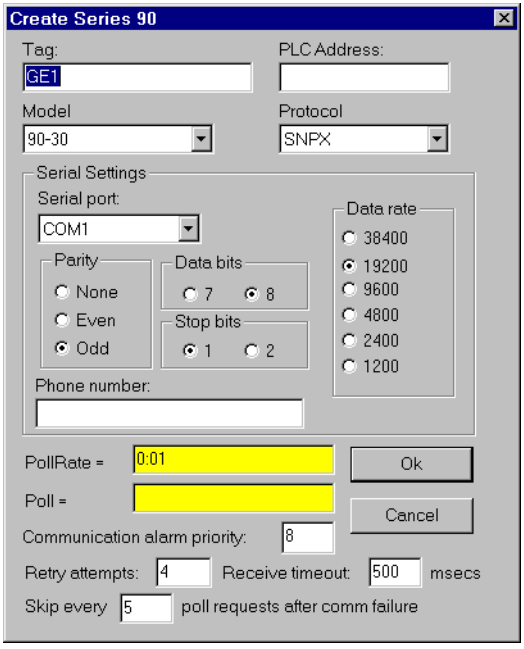


Figure 18-49. GE_Series90 Definition Parameters Dialog Box

PLC Address is a slave address and refers to the PLC address setting as configured on the device. The address can be up to eight ASCII characters.

Model chooses either 90-30 or 90-70.

Interface selects the protocol. Currently, only SNPX is supported.

Serial port specifies which port the object uses for communication to the external device. This does not specify the communication type. Communication type is determined by the **Options»Serial Ports…** command.

Data rate, **Parity**, **Data bits**, and **Stop bits** reference the settings on the hardware device.

Phone number specifies the number to be dialed if the serial port setting is configured for dial-up. This number only applies to the individual protocol object.

PollRate is a numeric expression that determines how often to poll the device. GE_Series90 then polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *Introduction*, for information on entering time constants.

Poll is a logical expression. When this expression changes from FALSE to TRUE, Lookout polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Communication alarm priority determines the priority level of object-generated alarms (0 – 10).

Retry attempts specifies the consecutive number of times Lookout attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the GE_Series90 object generates an alarm and releases the communication port back to the communications service which then moves on to the next device in the polling queue (if any). Refer to Chapter 6, *Serial Communications*, for more information.

Receive timeout is the time delay Lookout uses in waiting for a response from a device before retrying the request.

The **Skip every…** setting instructs Lookout to not poll a device it has lost communication with on every scheduled poll. Instead, Lookout skips the device in the polling cycle accordingly. Once communications have been reestablished, the device is polled on its regular cycle.

GE_Series90 Data Members

This protocol driver object contains a great deal of data. All readable and writable members (inputs/outputs), polling instructions, read/write blocking, serial port usage, and so on, are bundled with the object. Therefore, as soon as you create a GE_Series90 object you immediately have access to all the object data members (see data member list below).

☞ **Note** ***Lookout protocol driver objects automatically generate an efficient read/write blocking scheme based on the inputs and outputs being used in your process file. You are not required to build your own I/O blocking table.***

Table 18-50. GE_Series90 Data Members

Data Member	Type	Read	Write	Description
AI1 – AI64	numeric	yes	yes	16-bit analog inputs encoded as unsigned binary integers ranging from 0 to 65535
AID1 – AID64	numeric	yes	yes	32-bit analog inputs encoded as unsigned binary integers ranging from 0 to 65535
AQ1 – AQ64	numeric	yes	yes	16-bit analog outputs encoded as unsigned binary integers ranging from 0 to 65535
AQD1 – AQD64	numeric	yes	yes	32-bit analog outputs encoded as unsigned binary integers ranging from 0 to 65535
CommFail	logical	yes	no	Object-generated signal that is on if, for whatever reason, Lookout cannot communicate with the PLC.
I1 – I512	logical	yes	no	Single bit discrete inputs
OffHook	logical	no	yes	When TRUE, this flag instructs the GE object to retain exclusive use of its assigned communication port
Poll	logical	no	yes	When this value transitions from FALSE to TRUE, Lookout polls the device.
PollRate	numeric	no	yes	Lookout expression that determines the device polling frequency.
M1 – M4096	logical	yes	yes	Single bit discrete (Internal coil)
Q1 – Q512	logical	yes	yes	Single bit discrete outputs
R1 – R9999	numeric	yes	yes	16-bit holding registers encoded as unsigned binary integers ranging from 0 to 65535

Table 18-50. GE_Series90 Data Members (Continued)

Data Member	Type	Read	Write	Description
RD1 – RD9998	numeric	yes	yes	32-bit holding registers encoded as unsigned binary integers ranging from 0 to 65535
S1 – S32	logical	yes	no	System fault
SA1 – SA32	logical	yes	no	Special Contacts A
SB1 – SB32	logical	yes	no	Special Contacts B
SC1 – SC32	logical	yes	no	Special Contacts C
Update	logical	yes	no	Object-generated signal that pulses each time the driver polls the device

GE_Series90 Status Messages

No response within timeout period
Lookout did not received the expected response within the **Receive timeout** period. The object sent an inquiry and received an acknowledgment, but the device did not send an expected response to the request. This might happen if the response was interrupted. You may have to increase **Receive timeout**.

No return inquiry response from secondary unit
Lookout received no response from the device within the **Receive timeout** period. The driver object is able to use the COM port, but when it polls the device, it does not respond—as if it is not even there. You may have to increase **Receive timeout** to ensure Lookout is allowing enough time to receive the expected response. Also, verify your baud rate settings, cable connections, power, configuration settings, COM port settings, and polling addresses.

Bad LRC or BCC
The object is receiving a poll response from the device, but it could not decipher the response because it is garbled. Verify that all devices connected to the COM port have unique addresses. The last part of the message may actually be getting clipped off before it is completed. Consider increasing the number of **Retry attempts**. You may have to increase the **Receive gap** Serial Port setting to ensure Lookout is receiving the entire message. If your Serial Port is configured for radio, this could be caused by an audible squelch tail occurring at the end of a radio transmission. Try adjusting **RTS delay off** and **CTS timeout**.

No attach response within timeout period
An attempt was made to establish communications with the PLC without any response. Check your cabling and COM port selections, power, configuration settings, and polling addresses.

Invalid response [x]
An error in the structure of a response frame was detected. You may have two PLCs with the same address.

Incorrect response length [x]
A response was received with an unexpected length. You may have to increase the **Receive gap** Serial Port setting to ensure Lookout is receiving the entire message.

Incorrect response Address
A response was received with an address not matching the objects address. You may have two master devices on the network.

SNPX ERROR—Major code: x Minor code: x
The response message contained an SNPX error code. Refer to your GE documentation for the meaning of this particular error.

Histogram

The Histogram object class is one of the Lookout Statistical Process Control (SPC) tools and can play an important role in a Total Quality Management (TQM) program. This object class displays the distribution and/or relative distribution of a signal value. It shows the central tendency and variability of the sampled data. It also calculates process capability ratio (PCR), PCR confidence interval, mean, variance and standard deviation.

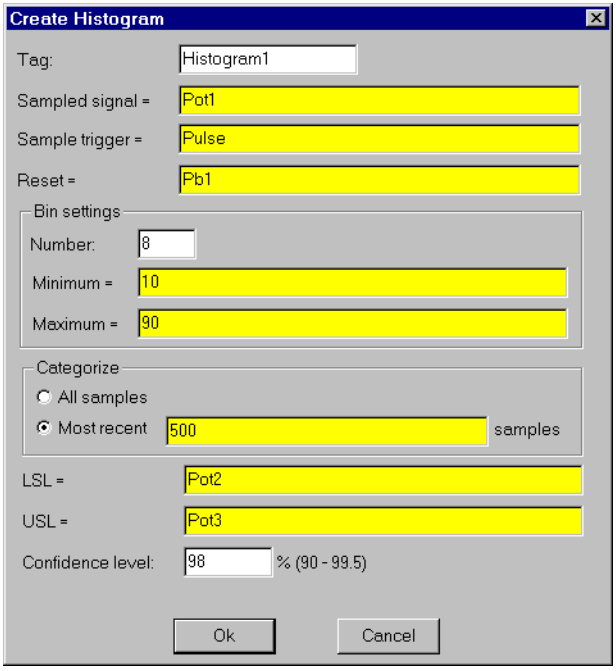


Figure 18-50. Histogram Definition Parameters Dialog Box

Sampled signal identifies the value that you are statistically monitoring. Histogram reads and categorizes the **Sampled signal** any time **Sample trigger** transitions from off to on.

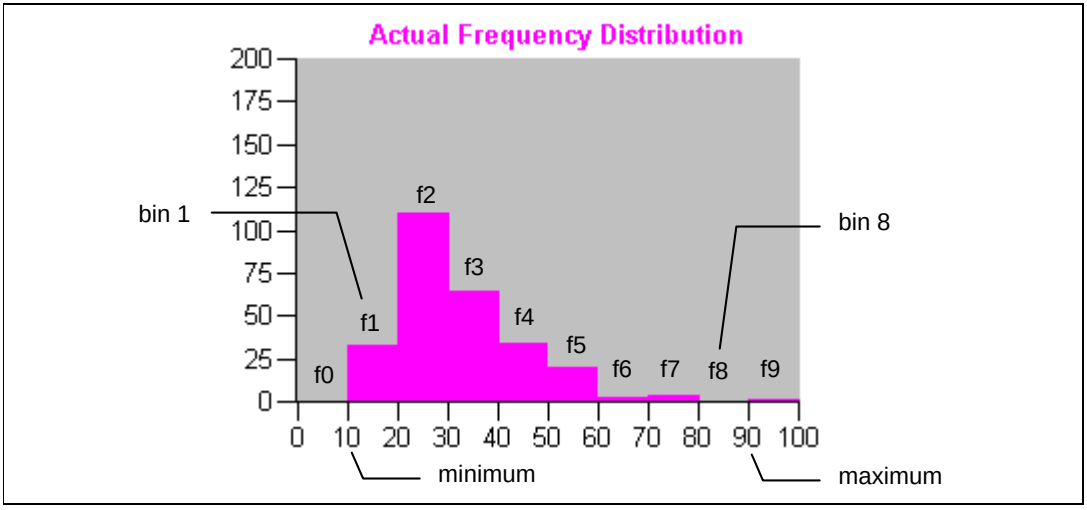
You can either **Categorize** all samples since **Reset** last transitioned from off to on, or you can **Categorize** a sliding window of the most recent 2 – 1000 samples.

Bin Settings define the **Number** of desired categories. The difference between **Minimum** and **Maximum** determines the width of each bin or category.

The settings for the Histogram graph below call for eight bins. Because the difference between **Minimum** and **Maximum** is 80 and because there are eight bins, each class interval has a range of 10 (80/8 bins = 10). So bin_1 contains sample values ranging from zero to less than 10; bin_2 contains sample values ranging from 10 to less than 20, and so on.

This object always allocates two more bins than you request. These bins are used for accumulating samples less than the **Minimum** and greater than the **Maximum**. The lower bin is always data member `f0`. The upper bin is `fn+1`, where `n`=**Number** in **Bin Settings**.

In the following example, Bin_0 (data member `f0`) contains all sample values less than **Minimum** and bin9 contains all sample values greater than or equal to **Maximum**.



This graph was created by inserting ten numeric expression barcharts (data members f0 through f9 as shown) and two scales.

LSL and **USL** identify the lower and upper specification limits. These limits generally define the range of acceptable sample values. Histogram uses **LSL** and **USL** to calculate the process capability ratios (`PCR` and `PCRk`).

The **Confidence level** expresses the degree of certainty or probability that the actual value of PCR (the process capability ratio) falls within the confidence interval (that is, between PCRL and PCRU). A typical value is 95 percent. The lower the confidence level, the tighter the interval.

☞ **Note** ***Histogram does not have a display parameters dialog box, However, you can easily display the results of its output signals by referencing them in expressions.***

Histogram Data Members

Table 18-51. Histogram Data Members

Data Member	Type	Read	Write	Description
f0 – f201	numeric	yes	no	Frequency of sampled signal values falling within the identified bin
LSL	numeric	yes	no	Lower specification limit
mean	numeric	yes	no	Average of sampled values, where $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$
PCR	numeric	yes	no	Process capability ratio measuring the uniformity or variability of the sampled signal using upper and lower specification limits. This ratio measures *potential* capability. If standard deviation = 0, PCR = 0. Otherwise $PCR = \frac{USL - LSL}{6\sigma}$
PCRk	numeric	yes	no	One-sided process capability ratio for an off-center process. This ratio takes process centering into account, measuring *actual* capability. $PCR_K = \mathbf{min}\left(\frac{\bar{x} - LSL}{3\sigma}, \frac{USL - \bar{x}}{3\sigma}\right)$
PCRL	numeric	yes	no	Lower limit of the confidence interval
PCRU	numeric	yes	no	Upper limit of the confidence interval

Table 18-51. Histogram Data Members (Continued)

Data Member	Type	Read	Write	Description
rf0 – rf201	numeric	yes	no	Relative frequency (percent) of sampled signal values falling within the identified bin
samples	numeric	yes	no	Number of sampled signal values,
sdev	numeric	yes	no	Sample standard deviation, where $\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$
USL	numeric	yes	no	Upper specification limit
variance	numeric	yes	no	Sample variance, where $\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$

Comments ***The confidence interval is the range of values determined by the limits PCRL and PCRU.***

Related Objects ***XBarR, Average, Maximum, Minimum, Sample***

Related Functions ***Avg, Max, Min, Stdev, Stdevp, Sum, Var, Varp***

Hitachi

Hitachi is a protocol driver class Lookout uses to communicate with Hitachi devices using the H series serial communication protocol.

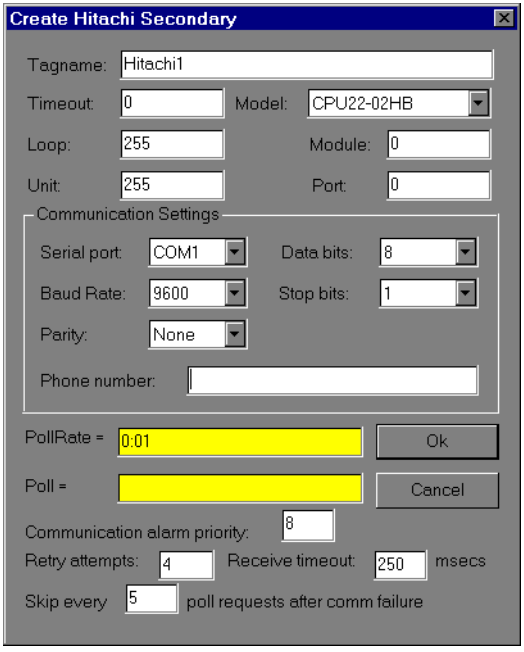


Figure 18-51. Hitachi Definition Parameters Dialog Box

Serial port specifies which comm. port the object uses for communicating to the external device. This does not specify the communication type. Communication type is determined by the **Options»Serial Ports…** command.

Data rate indicates the baud rate that Lookout uses to communicate with the hardware device. This setting should match the selection made on the physical device.

Data bits indicates the number of data bits that Lookout uses to communicate with the hardware device. This setting should match the selection made on the physical device.

Stop bits indicates the number of stop bits that Lookout uses to communicate with the hardware device. This setting should match the selection made on the physical device.

Parity indicates the parity that Lookout uses to communicate with the hardware device. This setting should match the selection made on the physical device.

Phone number specifies the number to be dialed if the selected serial port is configured for dial-up. This number only applies to the individual protocol object.

PollRate is a numeric expression that determines how often to poll the device. The object then polls the device at the specified time interval. Normally, this will be a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *Introduction*, for information on entering time constants.

When the logical value **Poll** transitions from FALSE to TRUE, Lookout polls the device. This can be a simple expression like the signal from a pushbutton, or it can be a complex algorithm.

Communication alarm priority determines the priority level of alarms generated by the Hitachi object. Such alarms are typically related to communications with the physical device.

Retry attempts specifies the consecutive number of times Lookout attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the Hitachi object generates an alarm and releases the communication port back to the communications subsystem, which then moves on to the next device in the polling queue (if any). Refer to Chapter 6, *Serial Communications*, for more information.

Receive timeout is the time delay Lookout uses in waiting for a response from a device before retrying the request.

The **Skip every…** setting instructs Lookout to not poll a device it has lost communication with on every scheduled poll. Instead, Lookout skips the device in the polling cycle. Once communications have been reestablished, the device is polled on its regular cycle.

Hitachi Data Members

An Hitachi object contains a great deal of data. It supports reading and writing of all predefined data points. When you create an Hitachi object, you have immediate access to all the object data members.

As with all Lookout drivers, you can access I/O points and other data through data members. The following is a table of data members currently supported by the Hitachi object class.

Table 18-52. Hitachi Data Members (Address Ranges in Hexadecimal)

Data Member	Type	Read	Write	Description
X0 – X200	logical	yes	yes	External bit input
WX0 – WX20	numeric	yes	yes	External word output
DX0 – DX10	numeric	yes	yes	External double-word output
Y0 – Y200	logical	yes	no	External bit output
WY0 – WY20	numeric	yes	no	External word output
DY0 – DY10	numeric	yes	no	External double-word output
R0 – R7C0	logical	yes	yes	Internal bit output
WR0 – WR7C	numeric	yes	yes	Internal word output
DR0 – DR3E	numeric	yes	yes	Internal double-word output
L0 – L4000	logical	yes	yes	Bit CPU link area
WL0 – WL2000	numeric	yes	yes	Word CPU link area
DL0 – DL1000	numeric	yes	yes	Double-word CPU link area
M0 – M4000	logical	yes	yes	Bit data area
WM0 – WM2000	numeric	yes	yes	Word data area
DM0 – DM1000	numeric	yes	yes	Double-word data area
TD0 – TD512	logical	yes	yes	On-Delay timer
CU0 – CU512	numeric	yes	yes	Up Counter
TC0 – TC512	numeric	yes	yes	Timer counter elapsed time
Update	logical	yes	no	Object-generated signal that pulses low each time it polls the device.
CommFail	logical	yes	no	Object-generated signal that is ON if Lookout cannot communicate with the device(s).

Table 18-52. Hitachi Data Members (Address Ranges in Hexadecimal) (Continued)

Data Member	Type	Read	Write	Description
Poll	logical	no	yes	When this expression transitions from FALSE to TRUE, Lookout polls the device.
PollRate	numeric	no	yes	Lookout expression that determines the device polling frequency.

Hitachi Status Messages

no acknowledgment for data request
no within timeout period
PLC response frame timeout
The PLC is not responding to data requests. Check communications settings, cables, and power.

Frame Error (Garbled): [Hitachi error code string]
Response frame was garbled. Check cabling and cable environment.

Hitachi frame error (RTC): [RTC error code string]
PLC received an invalid frame. Check communications settings.

Hitachi error message: [Hitachi error code string]
PLC received an invalid request. Check for a data member that is out of range for the model of PLC configuration.

HyperTrend

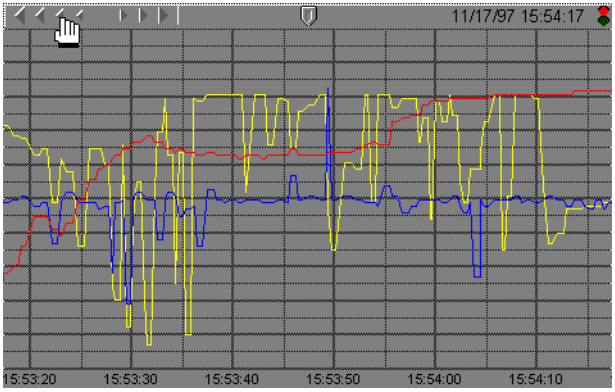
A HyperTrend object displays a trend graph on a control panel. It plots any number of logical and numeric trend lines.

HyperTrends provide instant access to both real-time and historical data in a single graph. For each plot line, they combine both real-time and historical data into a seamless, contiguous trace of data. See *Citadel Historical Database Logger* in Chapter 11, *Logging Data and Events*, for more information.

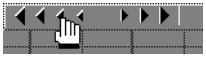
You can use HyperTrends to pan and zoom both the X axis and the Y axis, enabling dynamic adjustment of the vertical and horizontal resolutions of each plot line on the graph. Using this feature, you can, for example, zoom into a particular area of focus on the trend.

The graph scrolls from right to left, plotting current, real-time signals at the right end of the graph.

A button bar makes it easy for you to scroll the trend graph forward and back in time. It provides instant access to data that has scrolled off the left end of the graph (that is, historical data stored in the Citadel database).

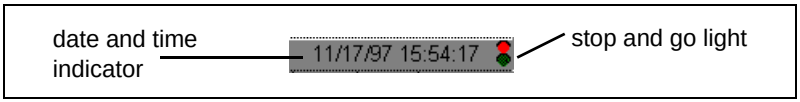


The button bar includes scroll arrows, a cursor button, date, time, and a stop & go light. Use the **scroll arrows** to move back and forth through time—the bigger the arrow button you select, the further the trend jumps in time. The scroll arrows also function much like a horizontal slider. Click on them and slide the mouse left and right while holding down the mouse button. The further you slide the cursor from dead center, the faster the trend scrolls in that direction.



Use the **date and time** indicators to choose a specific month, day, year, hour, minute, or second. If you click on the lower part of the hour, for example, it jumps back in time by one hour.

If you click on the upper part of the hour, it jumps ahead by one hour.



It works the same way for month, day, year, minute and second.

The **stop & go light** on the button bar is either red or green. If the light is green, it indicates the far right edge of the trend window displays the current time.

When you scroll back in time or if you click on the light when it is green, it changes to red, indicating that the trend is temporarily frozen in time. The date and time appears in the button bar indicating the exact time at the far right edge of the trend window. As you scroll back and forth through time, the data and time changes accordingly.

If you click on the light when it is red, the trend jumps back to current time and starts scrolling while plotting real-time values.

☞ **Note** ***The Citadel database continues to log data no matter what state the HyperTrend is in. You do not lose any data when it is in "historical" mode (that is, when the HyperTrend is not scrolling in real-time).***

When you click on the **cursor button**, a vertical cursor bar appears in the center of the graph along with an associated **Cursor** dialog box. The dialog box indicates the value of each trend line at the current location of the cursor. As you drag the cursor bar left and right on the trend graph, the values in the pop-up change to reflect the new cursor location.

You can select how the trend line values are shown by choosing a format through the dialog box control menu.

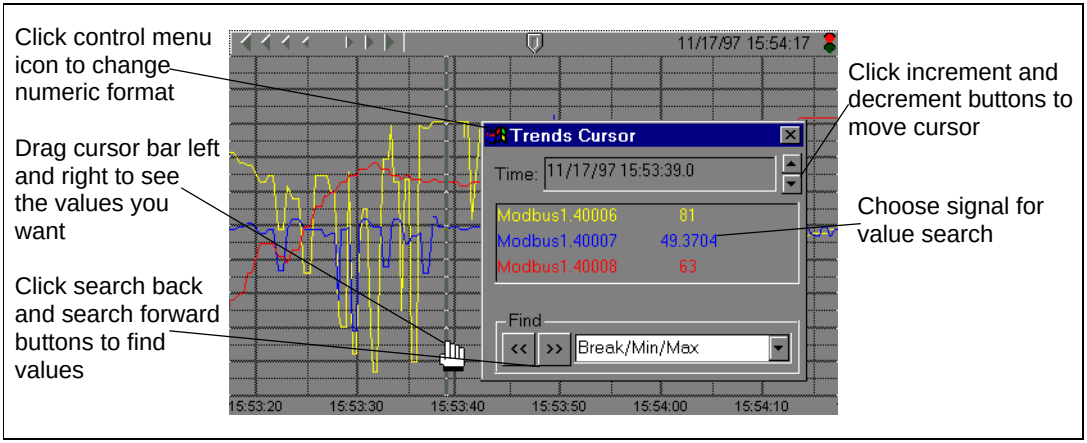


Figure 18-52. HyperTrend Cursor Dialog Box

Time indicates the current location of the cursor bar. The increment and decrement buttons beside the field move the cursor left and right in the trend graph. Choose the size of the incremental move by clicking on the desired portion of the date/time. The hour portion is selected in the example shown above, so each time you click on the increment or decrement button, the cursor bar jumps ahead or back by an hour. It works the same way for any portion of the date and time.

Use the **Find** combo box to search for a break in the trend line, a signal peak or valley, or a specific value. For example, you can find the last instance in which a process control limit value was exceeded. To find the last time a trend line crossed a specific value, choose the desired trend line by clicking on it in the list box, select `Value` in the **Find** combo box, enter the desired value, and clock on the scroll back button.

The HyperTrend definition dialog box is shown below.

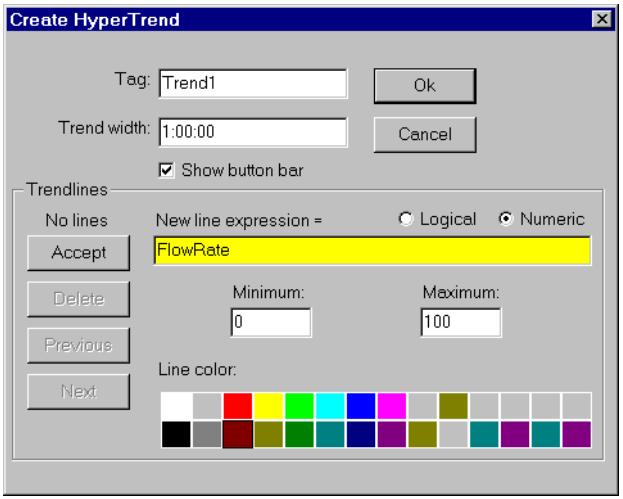


Figure 18-53. HyperTrend Dialog Box

Trend width is the default time span of the X axis on the graph. Graphs may have a default width, or time span, of anywhere from two seconds to four years. The default **Trend width** in the example dialog box indicates a time span of 1:00:00 or one hour. See *Numeric Data Members* in Chapter 5, *Developer Tour*, for more information on entering time constants. After creating the HyperTrend object, you can make the trend width adjustable by connecting a numeric signal to the `TrendWidth` data member.

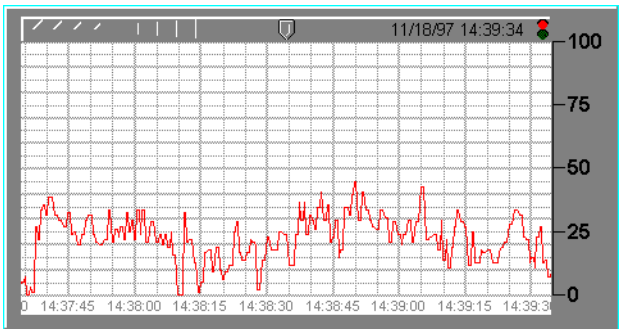
The **Show button bar** selection lets you hide or show the trend button bar on the control panel. You can make the button bar dynamically appear by connecting a logical signal to the `UseButtonBar` data member.

The **Trendlines** parameters enable adding, modifying, or deleting expressions from the trend graph. Typically, these expressions represent values that have previously been specified to be logged to the Citadel database. See Chapter 5, *Developer Tour*, and Chapter 11, *Logging Data and Events*, for logging data to the Citadel database. Enter logical and numeric expressions for plotting in the data field to the right of the Accept button. Choose **Logical** or **Numeric** to correspond with the current expression result.

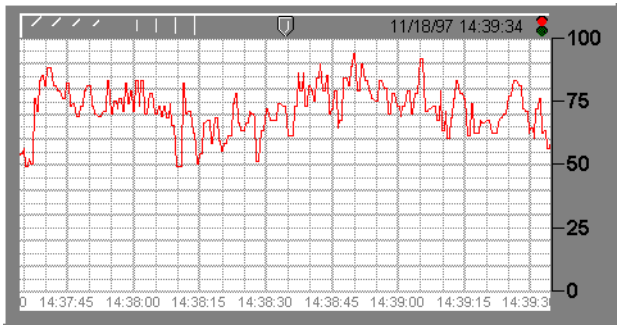
Line color specifies the color of the trend line for the current expression.

Minimum and **Maximum** settings determine where on the trend graph the expression is plotted. **Minimum** is the bottom of the graph while **Maximum** is the top of the graph—regardless of the range of the expression. These settings create an imaginary vertical scale and affect each expression independently.

For example, take two numeric expressions, both of which range from 0 to 50. Set the **Minimum** and **Maximum** to 0 and 100 on the first expression, and –50 and 50 on the second. The first expression plots in the bottom half of the chart while the second expression plots in the top half of the chart, even though they both fluctuate between 0 and 50.

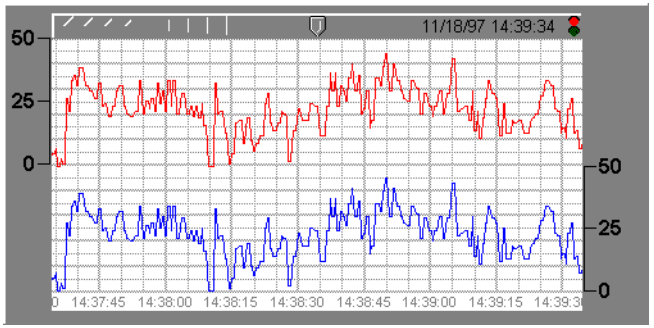


This figure shows the imaginary scale of the first expression (where min.=0 and max.=100). Because the expression ranges from 0 to 50, it is plotted in the bottom half of the graph.



This figure shows the imaginary scale of the second expression (where min.=–50 and max.=50). Because the expression ranges from 0 to 50, it is plotted in the top half of the graph.

When both expressions are entered on a single trend graph, you get the following effect. Notice the custom scales at either end of the graph.



If you select **Logical** for the expression type, the minimum and maximum settings changes to **Position** and **Height**. These two values now represent a number between 0% and 100%, and determine the baseline location of the trend line and its unit height when the expression goes TRUE.

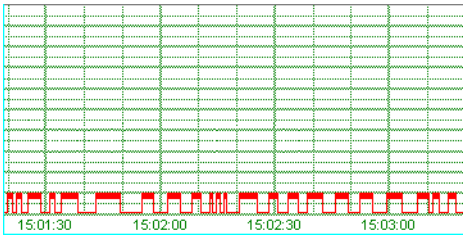


Figure 18-54. Plot of a Logical Value

When you finish entering or modifying the trend line parameters, click on the **Accept** button. This adds the expression to the **Trendlines** list. Use the **Delete** button to remove the current expression from the trend graph. The **Previous** and **Next** buttons toggle you through a list of all the expressions named for the current trend object.

Notice that each trend line is assigned a number. After creating the HyperTrend object, you can make the **Minimum** and **Maximum** settings of each numeric trend line and the **Position** and **Height** settings of each logical trend line adjustable by connecting numeric signals to the `Max`, `Min`, `Pos`, and `Height` data members. For example you can connect a pot to `Max1` to adjust the top of the graph for Trendline 1.

You can display HyperTrend graphs in various colors with different timeline styles and grid spacing, as shown on the following page.

Timeline labels determine where and how the date and time are to be displayed on the trend graph.

Major increments specifies the number of heavy horizontal grid lines on a trend graph. This value is independent of the range of any trend expressions.

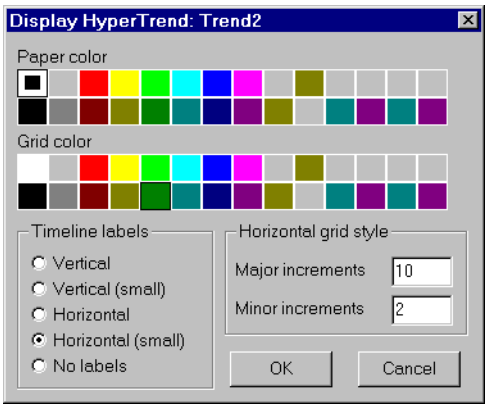


Figure 18-55. HyperTrend Display Parameters Dialog Box

Minor increments specifies the number of light horizontal grid lines between the major increment grid lines on a trend graph. This value is independent of the range of any trend expressions.

HyperTrend Data Members

Table 18-53. HyperTrend Data Members

Data Members	Type	Read	Write	Description
Enable1– Enable 999	logical	yes	yes	When TRUE, the identified trend line is visible. When FALSE, the trend line hidden. The default value is TRUE.
Height1 – Height999	numeric	yes	yes	Specifies the amplitude or height of the identified trend line (distance from baseline) when the logical expression goes TRUE. Height should be between 2 and (100 minus position).
Max1 – Max999	numeric	yes	yes	Specifies the top of the graph for the identified numeric trend line (the value of the trended line when it is at 100 percent of the Y axis).

Table 18-53. HyperTrend Data Members (Continued)

Data Members	Type	Read	Write	Description
Min1 – Min999	numeric	yes	yes	Specifies the bottom of the graph for the identified numeric trend line (the value of the trended line when it is at zero percent of the Y axis).
Pos1 – Pos999	numeric	yes	yes	Specifies the baseline location of the identified logical trend line. Baseline position should range should be 1–98. (pos1 is associated with trend line 1)
TrendWidth	numeric	yes	yes	Specifies the span of time that the X axis covers.
UseButtonBar	logical	yes	yes	When TRUE, the HyperTrend button bar becomes visible on the control panel. When FALSE, it is invisible. The default value is TRUE.
Visible	logical	yes	yes	When TRUE, the HyperTrend becomes visible on the control panel. When FALSE, it is invisible. The default value is TRUE.

Comments ***HyperTrend objects access data from the Citadel database. Think of them as windows into your historical database. If you have not specified a particular data point to be logged to the database, you can still reference its tagname in a HyperTrend object. When this is the case, Lookout automatically maintains the tagname in the Citadel database*** *for at least* ***twice the time span of the trend graph. For example, if your*** *trend width* ***is 1:00:00 (1 hour), Lookout automatically maintains*** *at least* ***the last two hours worth of data in the database—even if the tagname has not been previously specified to be stored to the Citadel.*** **See Chapter 5,** *Developer Tour***, and Chapter 11,** *Logging Data and Events***, for more information on specifying a point to be logged to the Citadel database.**

HyperTrends are updated as quickly as once per second, depending on screen resolution, the size of the graph, and the **trend width** setting. Computers with slow display adapters may slow down considerably when you display a large trend graph. On slower computers with slow display cards (no graphics coprocessor), consider limiting the size of your HyperTrends to less than one fourth the screen size.

You can display any number of trend lines on a given HyperTrend.

Integral

Integral is a totalizer—it totals the numeric **Input** signal. This class is typically used to total a measured flow rate.

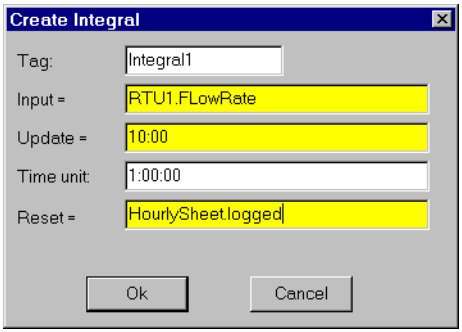


Figure 18-56. Integral Definition Parameters Dialog Box

Input is the numeric expression that you want to totalize or integrate.

Update can be a logical expression or numeric constant. If you specify **Update** as a numeric constant, it creates an internal Pulse timer with a pulse period of the specified time and a pulse duration of zero. See *Numeric Data Members* in See Chapter 5, *Developer Tour*, for information on entering time constants. If you specify **Update** as a logical variable, the variable should pulse at the desired frequency.

The **Update** expression extrapolates an interim total based on the current total and the most recent Input value. The interim total is then sent out as the output. The total is calculated using the trapezoidal numeric integration technique, and the total is corrected any time the incoming signal is refreshed.

Time unit is a numeric expression used as the basis for unit time on the Input signal. For instance, if the Input rate is in units of gallons per minute, the Time unit should be entered as one minute (1:00) so the totalized flow is in gallons. Typically the Time unit is one second (0:01), one minute (1:00), one hour (1:00:00), or one day (1:00:00:00). However, you can specify any unit, such as 5:23 (a rate of change in Input units per five minutes and 23 seconds).

Reset is a logical expression that resets the totalizer value to zero upon transition from OFF to ON.

☞ **Note** ***Integral does not have a display parameters dialog box. You can easily display the result of the Integral output signal by referencing its data member in an expression.***

Integral Data Members

Table 18-54. Integral Data Members

Data Member	Type	Read	Write	Description
(implicit)	numeric	yes	no	Totalized value

Comments ***The** Update **pulse forces the calculated total to continue changing between Input signal updates. For example, if a remote RTU that is monitoring a flow rate is polled every ten minutes, the** Update **pulse could be set at five seconds so the operator can watch the totalized flow continue to change on the screen as an extrapolated value. The corrected totalized value is calculated any time the** Input **signal refreshes—in this case, every ten minutes.***

If totalized values are logged to a spreadsheet on a daily basis, for example, and the total should be reset at the end of every day, use the update pulse generated by the Spreadsheet object to reset the total—this guarantees that the total is recorded before the totalizer is reset. The example on the previous page totalizes the hourly flow for permanent data logging by a spreadsheet object named HourlySheet. Notice that the spreadsheet update pulse `HourlySheet.logged` is used to reset the totalizer.

Related Objects ***Accumulator, Counter, Derivative***

Interpolate

The Interpolate object class performs a linear interpolation between a set of X,Y coordinate pairs to determine a corresponding output for any given input. A single Interpolate object can have up to 1,000 coordinate pairs. Although this object class can be used for any calculation requiring linear interpolation, it is especially useful for tank strapping applications.

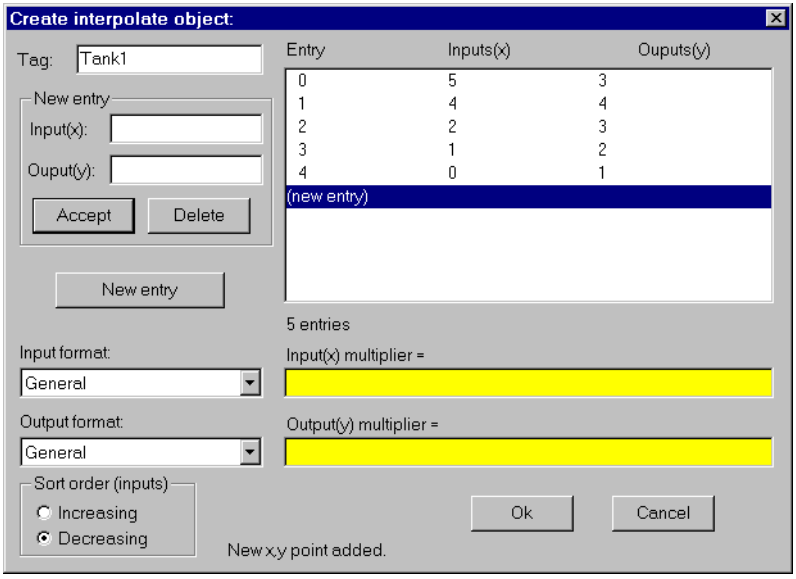
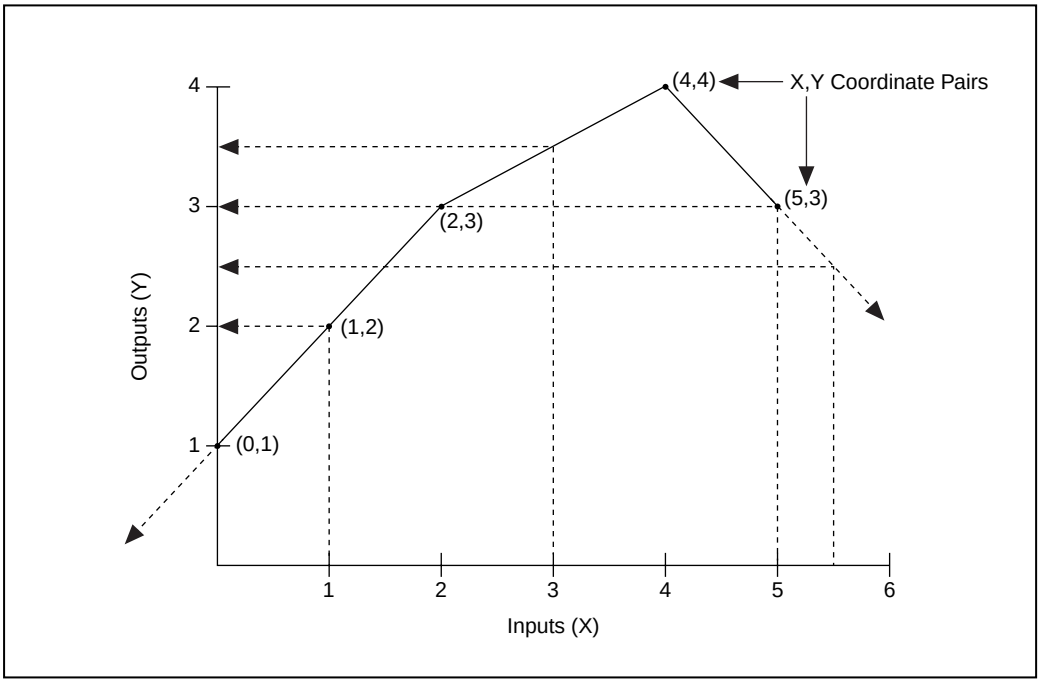


Figure 18-57. Interpolate Definition Parameters Dialog Box

The following diagram visually depicts the basic functionality of the Interpolate object, and the set of coordinate pairs as entered above.

Based on this graphical representation of the object, you can readily see how different X inputs give corresponding Y outputs. Also notice that an X input of 5.5 yields a Y output of 2.5—even though the last coordinate pair was (5,3). Lookout uses straight line extrapolation at both end points with a slope determined by the last two coordinate pairs.



The **New entry** parameter fields enable adding, modifying and deleting **Input(x)** and **Output(y)** coordinate pairs. **Input(x)** and **Output(y)** are numeric parameters.

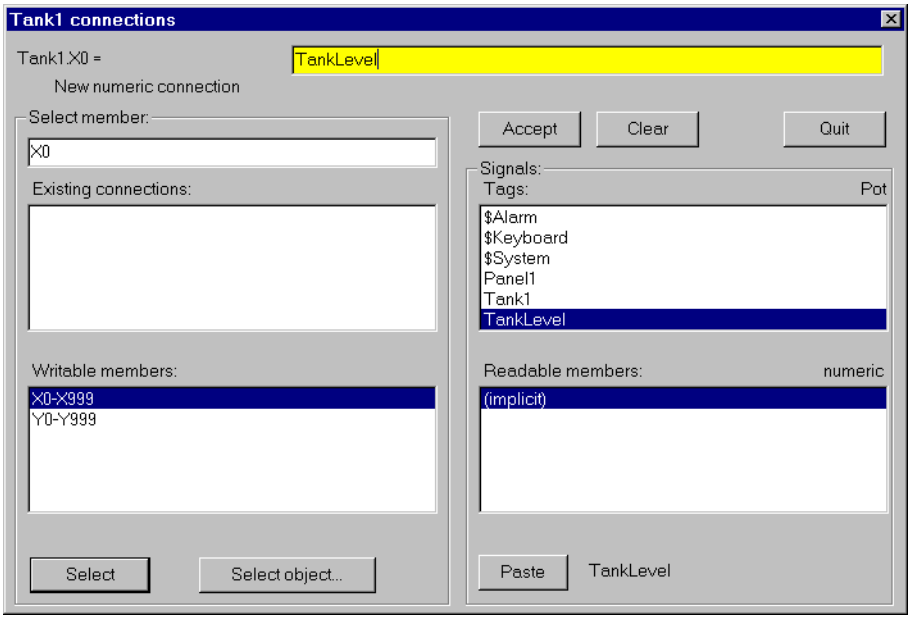
Input format and **Output format** define the numeric format of the **Input(x)** and **Output(y)** coordinate pairs displayed in the definition dialog box.

Sort order (inputs) specifies whether the dialog box lists X,Y coordinate pairs in **Increasing** or **Decreasing** order. The sort order has no bearing on output calculations or the values of the readable data members.

The **Input(x) multiplier** and the **Output(y) multiplier** are numeric expressions. These two parameters are useful for the more advanced calculations often found in tank strapping applications. They can be used for such things as temperature, pressure, density, and product correction factors. As their names imply, Lookout multiplies the corresponding Input/Output by the respective multiplier. If you specify an **Input(x) multiplier**, the object multiplies the input by the appropriate multiplier *before* calculating an interpolated output. If you specify an **Output(y) multiplier**, the object multiplies an interim output by the appropriate multiplier *before* calculating the final output value.

Step through a simple example using the diagram above to clarify this concept. Use an **Input(x) multiplier** of 1.5, and a **Output(y) multiplier** of 2. Assume the variable input is currently 2. The object first multiplies 2 by 1.5 to produce 3. It then uses the coordinate pairs to find the appropriate output value for an input of 3. As you can see, the interim output would be 3.5. Because you specified an output multiplier, it must first multiply 3.5 by 2 to give a final output of 7.

Situations may arise where you want multiple inputs going through a single Interpolate object, giving multiple respective outputs. Instead of creating an Interpolate object for each input, Lookout can connect multiple inputs to a single object and read their corresponding outputs. The inputs are writable data members (X0 – X999). For each input, there is a corresponding readable output (Y0 – Y999 respectively). The diagram below shows a numeric signal called `TankLevel` connected to an input, X0.



The Interpolate object now multiplies the X0 input by the **Input(x) multiplier**, find the two coordinate pair parameters that X0 falls between, interpolate, and multiply the interim output by the **Output(y) multiplier**, then send the resulting value to the Y0 readable data member.

Note ***The Interpolate object class does not have a display parameters dialog box. However, you can easily display the result of its output signals by referencing its data members in expressions.***

Interpolate Data Members

Table 18-55. Interpolate Data Members

Data Members	Type	Read	Write	Description
X0 – X999	numeric	yes	yes	Input value
Y0 – Y999	numeric	yes	yes	Output value. For a given input, X*n*, the object outputs the result of the interpolation as a corresponding Y*n* value.

Comments ***If you want to interpolate another input using the same coordinate pair parameters and multipliers (that is, the same Interpolate object), connect to another input like X23, and read its corresponding output, Y23.***

Because X*n* and Y*n* data members are both readable and writable, the object can also act as a bi-directional interpolator. For example, you can connect a numeric signal to a Y*n* data member, and read the corresponding interpolated X*n* data member. A real-world example of this could be a typical tank strapping problem. Normally, you would use a tank level as an input to X0, and read the interpolated output Y0 as the corresponding tank volume. Hence you begin with a level and end up with a volume. However, you might also be monitoring the volume and would like to use that value to calculate a corresponding level. In that case, you might want the Interpolate object to be bi-directional, which it is. If Y0 ever changes, the object *divides* Y0 by the **Output(y) multiplier**, interpolate between the two closest **Output(y)** parameters, calculate a corresponding input value based on the two closest **Input(x)** parameters, divide it by the **Input(x) multiplier**, and send the resulting value to the X0 data member.

Interval

Interval is an adjustable delay timer. When **On/off signal** transitions to on, its output turns on and the **Timer delay** begins to count down. At the end of the delay countdown, the output signal turns OFF. If **On/off signal** is dropped at any time, the output signal turns OFF, and the timer is reset.

Timer delay can range from 0.0 seconds to several years, and the effective resolution is 0.1 seconds over the entire range. The timer display digitally shows the time delay remaining. It is updated approximately once per second. If the **On/off signal** is low, or the time delay period has expired, the timer display shows `OFF`.

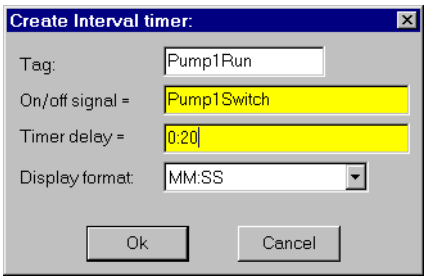


Figure 18-58. Interval Definition Parameters Dialog Box

The **On/off signal** is a logical expression while **Timer delay** is a numeric expression. Normally, this is a simple time constant such as 0:20 (twenty seconds). See *Numeric Data Members* in See Chapter 5, *Developer Tour*, for information on entering time constants.

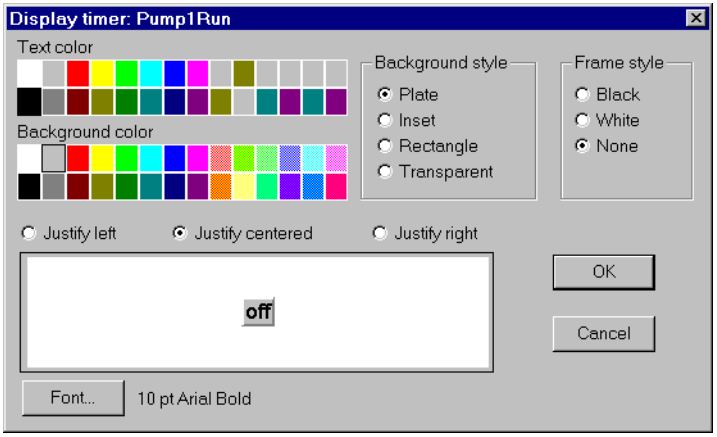


Figure 18-59. Interval Display Parameters Dialog Box

Interval Data Members

Table 18-56. Interval Data Members

Data Member	Type	Read	Write	Description
(implicit)	logical	yes	no	Logical timer value

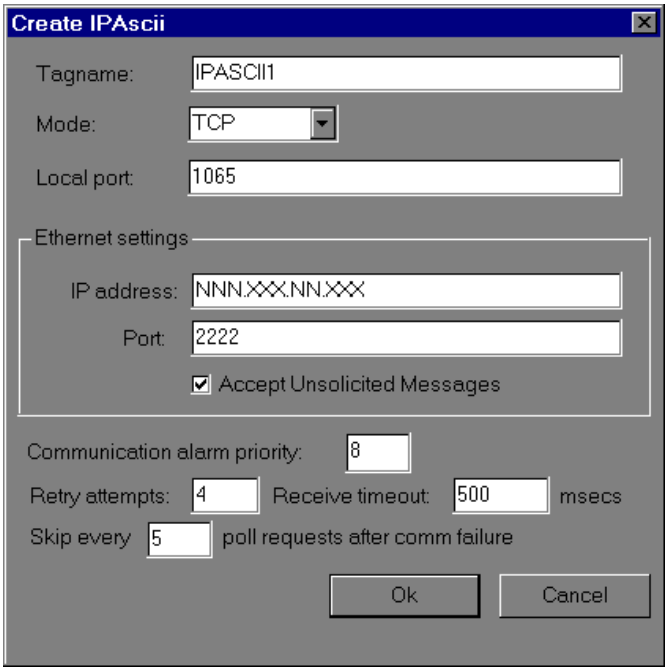
Comments ***The Interval timer can be used to enforce a maximum run time for a pump.***

Related Objects ***DelayOff, DelayOn, OneShot, Pulse, TimeOfxxx***

IPASCII

IPASCII is a protocol driver class Lookout uses to communicate with any IP device that runs standard TCP or UDP services.

An IPASCII object contains no predefined data points. When you create a IPASCII object, you must define your data request strings as well as the template Lookout uses to parse the response frame.



Local port specifies which local Ethernet I/O port the object uses for communicating to the external device.

Mode indicates whether the object will use a TCP socket or a UDP socket.

IP address indicates the address of the device you wish to communicate with.

Port indicates the Ethernet I/O port number on the remote device where the object will try to establish a socket.

Accept Unsolicited Messages indicates whether or not the object will report packets from the device that are not in response to a request that the

IPASCII object has sent it. This does *not* mean that the IPASCII object can act as a server. It means that once the IPASCII object has established a socket as a client, it reports unexpected data it receives from that point on.

Communication alarm priority determines the priority level of alarms generated by the IPASCII object. Such alarms are typically related to communications with the physical device.

Retry attempts specifies the number of times Lookout attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the IPASCII object generates an alarm and releases the socket.

Receive timeout is the amount of time Lookout waits for a response from a device before retrying the request.

The **Skip every ___ poll requests after comm failure** setting instructs Lookout not to poll a device it has lost communication with on every scheduled poll. Instead, Lookout polls the device only once in the specified number of poll cycles. Once communication has been reestablished, the device is polled on its regular cycle. In the context of the IPASCII object, this means that this number of Send commands will be ignored until communication has been reestablished.

IPASCII Data Members

Table 18-57. IPASCII Data Members

Data Member	Type	Read	Write	Description
RequestFormat	text	no	yes	Format used to create request frame.
ResponseFormat	text	no	yes	Format used to parse response frame.
Send	logical	no	yes	Sends request frame.
RQV1—RQV100	numeric	no	yes	Variable list used to populate request frame with numeric values.
RQV1.txt—RQV100.txt	text	no	yes	Variable list used to populate request frame with text values.
RQV1.logical— RQV100.logical—	logical	no	yes	Variable list used to populate request frame with logical values.
RSV1—RSV100	numeric	yes	no	Variable list used to store values retrieved from response frame.

Table 18-57. IPASCII Data Members (Continued)

Data Member	Type	Read	Write	Description
RSV1.txt— RSV100.txt	text	yes	no	Variable list used to store values retrieved from response frame.
RSV1.logical—RSV100.logical	logical	yes	no	Variable list used to store values retrieved from response frame.
Request	text	yes	no	Exact request frame sent.
Response	text	yes	no	Exact response frame received.
CommFail	logical	yes	no	Object-generated signal that is on if Lookout cannot communicate with the device(s).
Update	logical	yes	no	Object-generated signal that pulses low each time it polls the device.

* RQVn, RQVn.txt and RQVn.logical all represent the same value in different forms.

* RSVn, RSVn.txt and RSVn.logical all represent the same value in different forms.

Request and Response Format Strings

The request and response format strings consist of static characters and markers that control how the request and response frames respectively are formatted or decoded. The request format string is used to *create* the request frame, which is sent to the device, while the response format string is used to *decode* the response frame, which comes from the device.

Static characters in the format strings are reproduced exactly in the request or response frame. Markers specify the location within the frame and type of data which should be found there, such as five characters read as an unsigned integer, for example. The IPASCII object constructs a request frame by processing the sequence of static characters and markers in the request format string, and including data from RQV data members.

The response format string decodes a response frame using an analogous process, storing the results in RSV data members.

To construct a request frame, the IPASCII object parses the request format string character by character. Static characters are copied directly to the request frame. When a marker is encountered the IPASCII object reads a value from the appropriate RQV variable and places it into the request frame.

There are 100 RQV and RSV values provided for in the IPASCII object data member collection. The first marker in a format string uses the value from `RQV1` (or `RQV1.txt` or `RQV1.logical`), the next marker uses the value `RQV2`, and so on. Values taken from Response strings are stored in RSV data members in the same way.

Keep in mind that writing into `RQV1` changes the value both for `RQV1.text` and `RQV1.logical`. Their only difference is the format in which they are represented. The same principle applies to the RSV data members.

☞ **Note** ***There is no precedence to the order in which multiple objects connected to the same variable number will initialize upon opening the process file, such as the case in which a Pot object is connected to RQV1 while a TextEntry object is connected to RQV1.txt. You should take care to initialize such variables to the proper value after opening a process file.***

To decode a response frame, the IPASCII object compares the response frame to the response format string character by character. The static characters in the response frame must match those in the response format string or the decoding process terminates. Static characters are, in effect, discarded by the IPASCII object as they are matched between the response format string and the response frame.

When the IPASCII object encounters a marker, it places the data indicated by the marker into the appropriate RSV data member.

The conversion of a portion of the response frame to a data type specified by a marker in the response format string must be valid, or the process will terminate.

If nothing halts the process, decoding terminates when the end of the response frame string is reached.

There are examples of both request frames and response frames at the end of this section, but for the examples to make sense, you must first understand the IPASCII object markers.

Markers

The general format for a marker is:

%[width][type]

Each field in the marker format is a single character or a number signifying a particular format option.

The **%** sign denotes the beginning of the marker. If the percent sign is followed by a character that has no meaning as a format-control character, that character and the following characters (up to the next percent sign) are treated as static characters, that is, a sequence of characters that must match the frame exactly. For example, to specify that a percent-sign character is a static character part of the frame, use %%.

Width is a positive decimal integer specifying the number of characters that particular value occupies in the frame. By default IPASCII will pad the value with blank spaces if the value takes up fewer characters than the value specified by width. Including a 0 before the width value forces the IPASCII object to pad with zeroes instead of blank spaces.

Type determines whether the field is interpreted as a character, a string, or a number.

Table 18-58. Data Types Allowed by IPASCII

Character	Data Type
d	Decimal integer
x, X	Hexadecimal integer
u	Unsigned decimal integer
f	Floating-point
s	String
b	Byte (binary)

The simplest format specification contains only the percent sign and a type character (for example, %s). That would place the value in the response frame in the RSV1.txt data member.

Request Format String	RQV1	Request Frame
>%5d	34	> 34
>%05d	34	>00034

The request format string also has a precision value in the form **%[width].[precision][type]**. This specifies the number of digits to the right of the decimal point, if any, in the request frame. If you use a float

(%f) and do not specify a precision value, the IPASCII object assumes a default of 6.

Characters are converted and stored in RSV data members from response frames in the order they are encountered in the response format. However, fewer than **[width]** characters may be read if a white-space character (space, tab, or newline) or a character that cannot be converted according to the given format occurs before **[width]** is reached.

Values needed for request frames come from the RQV data members, and are also used in the order in which they occur in the request format.

To read strings not delimited by space characters, or that contain spaces, you can substitute a set of characters in brackets (**[ ]**) **s** (string) type character. The corresponding input field is read up to the first character that does not appear in the bracketed character set. Using a caret (^) as the first character in the set reverses this effect: the IPASCII object reads input field up to the first character that does appear in the rest of the character set.

Response Format String	RSV1.txt	Response Frame
$[A – Z,a – z,]$	Natl Inst	$Natl Inst$
>[^,s]	days	>day

Notice that **%[a-z]** and **%[z-a]** are interpreted as equivalent to **%[abcde...z]**, and that the character set is case sensitive.

☞ **Note** ***The brackets only work in response format strings. They have no effect in the request format string.***

The IPASCII object scans each field in the response frame character by character. It may stop reading a particular field before it reaches a character for a variety of reasons:

- The specified width has been reached.
- The next character cannot be converted as specified.
- The next character conflicts with a character in the response format string that it is supposed to match.
- The next character fails to appear in a given character set.

No matter what the reason, when the IPASCII object stops reading an field, the next field is considered to begin at the first unread character. The

conflicting character, if there is one, is considered unread and is the first character of the next field.

Entering the Format String

For a static connection to one of the format data members, enter your format string in the yellow field box in the Edit Connections dialog box. Remember to begin and end the format strings with quotation marks so that Lookout will accept the string input.

You can also connect any valid text data member, such as a text entry object, to the format data members.

Request Frame Construction Examples

Request Format String	RQV	Request Frame
<01%4u%s	RQV1=1234 RQV2.txt=Ross	<011234Ross
<01%04u%s	RQV1=34 RQV2.txt=Ross	<010034Ross
<01% 4u%s	RQV1=34 RQV2.txt=Ross	<01 34Ross

A zero in front of the four pads with zeroes; a space pads with spaces.

Response Format Examples

Response Frame	Response Format String	RSV
*(16.38:	*(%5.2f:	RSV1=16.38

☞ **Note** ***The decimal point counts as a character when decoding floats (%f). Also, decimal points denoting precision are not allowed when decoding a float in the response frame.***

>>Test Text<<	>>%s<<	RSV1.txt=Test

The space between the words terminates the conversion. See the bracketed character example above in order to span a space or other special characters. The response format uses a space as a delimiter.

>>Test Text<<	>>%s%s<<	RSV1.txt=Test RSV2.txt=Text
>>DogCat<<	>>%3s%3s<<	RSV1.txt=Dog RSV2.txt=Cat

IPASCII Error Messages

No response from device within timeout period
Lookout received no response from the device within the **Receive timeout** period. The IPASCII object was able to establish a socket, but when it sends its message to the device, it does not respond—as if it is not even there. You may have to significantly increase **Receive timeout** (and **Poll Rate**) to ensure Lookout is allowing enough time to receive the expected response. Also, verify your cable connections, power, configuration settings, and IP settings.

Not enough data to send a valid frame
This means that the IPASCII object has not received enough data to fill in all the variables in the Request Format frame. This could mean that you do not have connections made to all of the RQVs that the IPASCII object is expecting.

Socket communications error
This alarm message will be followed by a Windows standard socket error message. The most common reason you might see one of these errors is if an error has occurred on the socket after a valid socket has already been established.

Cannot communicate with device
This means that the IPASCII object was not able to establish a valid socket with the remote device, and in fact did not get any kind of response from the device.

Cannot resolve IP address
Check to make sure you have given IPASCII a valid IP address.

Cannot resolve port
Check to make sure you have given IPASCII a valid port number on the remote device.

Garbled or unexpected response
IPASCII got a response frame, but static characters in the response did not match up to the response format string.

Illegally formatted string received
IPASCII got a response frame, but certain characters in the response were not in the format stated by the markers in the response format string.

Related Objects ***ASCII***

Junction

Junction receives up to nine numeric values, each of which could be the result of a complex numeric expression. It outputs the value of the last **Additional input** that changed (event driven). Notice, however, that it will not output a value until the **Initializing input** has changed. After that, any change in any input is immediately output.

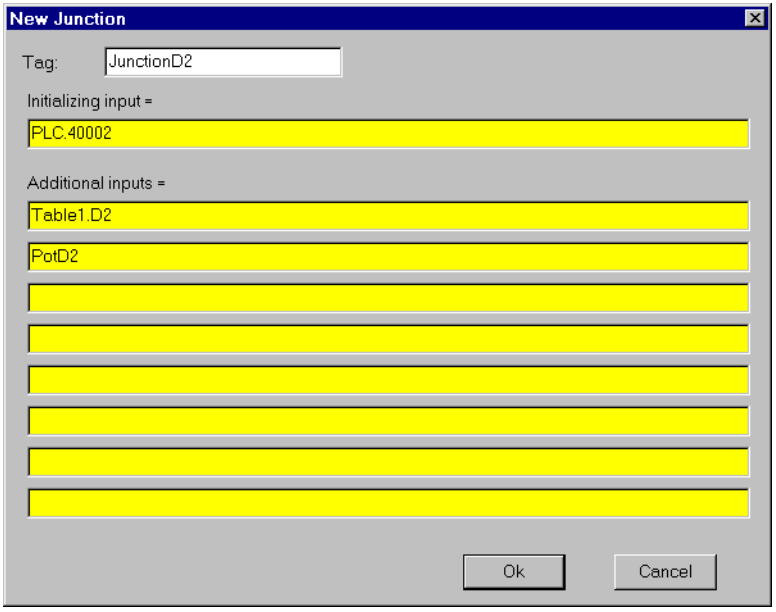


Figure 18-60. Junction Definition Parameters Dialog Box

Junction Data Members

Table 18-59. Junction Data Members

Data Member	Type	Read	Write	Description
(implicit)	numeric	yes	no	The value of the input that most recently changed

Comments ***Junction is a unique class that should be used only in rare circumstances.***

$Keyboard

$Keyboard is a global object. Its data members represent the keyboard function keys. Unlike other object classes in which you can create several objects of the same class, you cannot create or delete $Keyboard objects, but you can use the one supplied.

You can use the $Keyboard global object to perform such functions as calling a particular control panel, activating a batch sequence, or acknowledging alarms by pressing a key.

Think of $Keyboard data members (which represent function keys on the keyboard) as Lookout pushbuttons. Just as you can connect a pushbutton to the activate data member of a panel, you can also connect a $Keyboard data member to the activate data member of a panel. Such a connection is shown below:

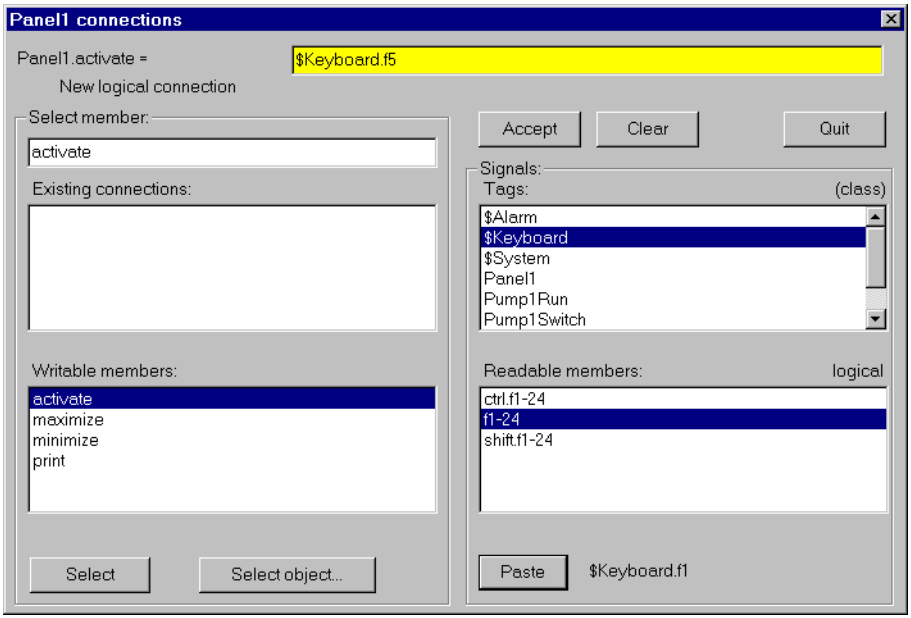


Figure 18-61. Edit Connections Dialog Box

The logical expression, `$Keyboard.F1` calls up Panel1 any time a user presses the F1 key on the keyboard. Similar connections could be made to other panels. You can easily connect a different panel to each function key.

Just as easily, you can connect a function key to a batch process trigger. When the key is pressed, (that is, when the `$Keyboard` data member goes TRUE) the batch is activated—reading batch ingredients from a recipe object, opening and closing valves, starting mixers, bottling finished material, and so on.

You might also connect a function key to `$Alarm.ack`. This would enable users to acknowledge alarms through a single keystroke.

$Keyboard Data Members

$Keyboard has 72 readable data members. Each data member represents a unique key sequence, described in the following table.

Table 18-60. $Keyboard Data Members

Data Member	Type	Read	Write	Description
F1 - F24	logical	yes	no	Each of these 24 data members represent a function key, F1 – F24. A given data member returns a logical TRUE when its associated function key is pressed and FALSE when the key is released.
Shift-F1 - Shift-F24	logical	yes	no	Each of these 24 data members represent a function key, F1 – F24—when pressed in conjunction with the Shift key. A given data member returns logical TRUE when the Shift key and function key are pressed together and FALSE when the keys are released.
Ctrl-F1 - Ctrl-F24	logical	yes	no	Each of these 24 data members represent a function key, F1 – F24—when pressed in conjunction with the Ctrl key. A given data member returns logical TRUE when the Ctrl key and function key are pressed together and FALSE when the keys are released.

Comments ***$Keyboard function keys are global in nature. Any time F1 is pressed, the*** `$keyboard.F1` ***signal goes TRUE—regardless of what panel the user is looking at. If you want a function key to be unique from one control panel to the next, use the Panel object class function key data member. See*** *Panel* ***object class definition for more information.***

Related Objects ***Pushbutton***

LatchGate

LatchGate is latched on and off by two incoming signals. It retains the state of the signal that most recently went high, regardless of the state of the other signal. When the **Turn Off** signal transitions from OFF to ON, the LatchGate output goes OFF until the **Turn On** signal transitions from OFF to ON. The output signal does not change when either incoming signal transitions from ON to OFF. Both **Turn Off** and **Turn On** are logical expressions.

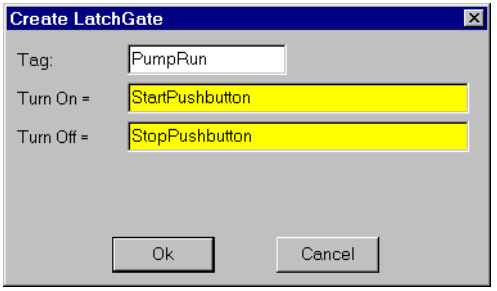


Figure 18-62. LatchGate Definition Parameters Dialog Box

LatchGate Data Members

Table 18-61. LatchGate Data Members

Data Member	Type	Read	Write	Description
(implicit)	logical	yes	no	Logical output signal value

Comments ***Two pushbuttons connected to the*** *Turn On* ***and*** *Turn Off* ***expressions of a LatchGate create pushbutton start/stop controls for a pump or other device.***

Related Objects ***Flipflop***

Maximum

Maximum actively calculates the maximum value of **Data** over time. Maximum is only active when the **Enable** expression is TRUE. It resets to zero when the **Reset** expression transitions from OFF to ON. Maximum also maintains an array of up to 35 previous maximum values. If **Enable** is left blank, the object always actively calculates the maximum. **Data** is a numeric expression while **Reset** and **Enable** are logical expressions.

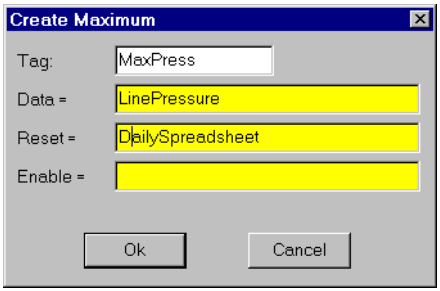


Figure 18-63. Maximum Configuration Parameters Dialog Box

☞ **Note** ***Maximum does not have a display parameters dialog box. However, you can easily display Maximum by referencing it in an expression.***

Maximum Data Members

Table 18-62. Maximum Data Members

Data Members	Type	Read	Write	Description
(implicit)	numeric	yes	no	Current maximum value
1 – 35	numeric	yes	no	Previous maximum values. Signal 1 is the most recent prior maximum since the Reset expression went high.
DataReset	logical	no	yes	Upon transition from FALSE to TRUE, resets to zero all data members—including the current maximum value and all previous maximum values.

Comments ***The Reset interval could be a regular pulse interval created by a TimeOfxxxx timer, so that the pulse is synchronized to the top of the hour or day. For example, if you want to calculate the daily maximum flow rate, use the***

output signal from a TimeOfDay timer or a daily Spreadsheet object to reset the maximum calculation at the beginning of each day.

Related Objects ***Minimum, Average, Sample, SampleText***

 © National Instruments Corporation

Minimum

Minimum actively calculates the minimum level of **Data** over time. Minimum is only active when the **Enable** expression is TRUE. It resets to zero when the **Reset** expression transitions from OFF to ON. Minimum also maintains an array of up to 35 previous minimum values. If **Enable** is left blank, the object is always actively calculating the minimum. **Data** is a numeric expression while **Reset** and **Enable** are logical expressions.

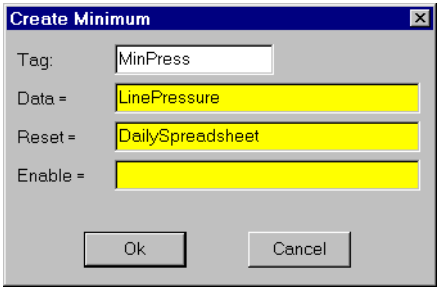


Figure 18-64. Minimum Configuration Parameters Dialog Box

Note ***Minimum does not have a display parameters dialog box. You can easily display the Minimum value referencing it an expression.***

Minimum Data Members

Table 18-63. Minimum Data Members

Data Members	Type	Read	Write	Description
(implicit)	numeric	yes	no	Current minimum value
1 – 35	numeric	yes	no	Previous minimum values. Signal 1 is the most recent prior minimum since the Reset expression went high.
DataReset	logical	no	yes	Upon transition from FALSE to TRUE, resets to zero all data members—including the current minimum value and all previous minimum values.

Comments ***The Reset interval could be a regular pulse interval created by a TimeOfxxxx timer, so that the pulse is synchronized to the top of the hour or day. For example, if you want to calculate the daily minimum flow rate, use the output***

signal from a TimeOfDay timer or a daily Spreadsheet object to reset the minimum calculation at the beginning of each day.

Related Objects ***Maximum, Average, Sample, SampleText***

Mitsubishi
MitsubishiFX

Mitsubishi is a protocol driver class Lookout uses to communicate with Mitsubishi devices using the serial communication protocol.

A Mitsubishi object contains a great deal of data. It supports reading and writing of all predefined data points. When you create a Mitsubishi object, you have immediate access to all the data members for that object (see *data member* list below).

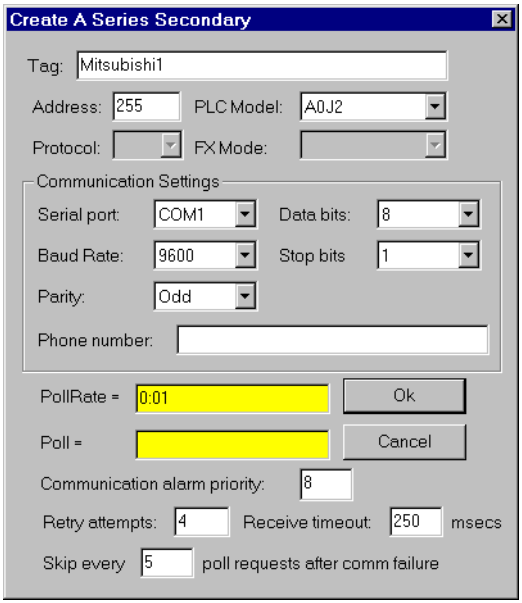


Figure 18-65. Mitsubishi Configuration Parameters Dialog Box

Serial port specifies which COM port the object uses for communicating to the external device. This does not specify the communication type. Communication type is determined by the **Options»Serial Ports…** command.

Data rate indicates the baud rate that Lookout uses to communicate with the hardware device. This **Data rate** setting should match the selection made on the physical device.

Data bits indicates the number of data bits that Lookout uses to communicate with the hardware device. This **Data bits** setting should match the selection made on the physical device.

Stop bits indicates the number of stop bits that Lookout uses to communicate with the hardware device. This **Stop bits** setting should match the selection made on the physical device.

Parity indicates the parity that Lookout uses to communicate with the hardware device. This **Parity** setting should match the selection made on the physical device.

Phone number specifies the number to be dialed if the selected serial port is configured for dial-up. This **Phone number** only applies to the individual protocol object.

PollRate is a numeric expression that determines how often to poll the device. The object then polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *Introduction*, for more information on entering time constants.

Poll is a logical expression. When this expression changes from FALSE to TRUE, Lookout polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Communication alarm priority determines the priority level of alarms generated by the Mitsubishi object. Such alarms are typically related to communications with the physical device.

Retry attempts specifies the consecutive number of times Lookout attempts to establish communications with a device when it is not getting a valid response. After it tries the number of **Retry attempts** specified, the Mitsubishi object generates an alarm and releases the communication port back to the communications subsystem. The subsystem then moves on to the next device in the polling queue (if any). See Chapter 6, *Serial Communications*, for more information.

Receive timeout is the time delay Lookout uses in waiting for a response from a device before retrying the request.

The **Skip every…** setting instructs Lookout not to poll a device it has lost communication with on every scheduled poll. Instead, Lookout skips the device in the polling cycle. Once communications have been reestablished, the device is polled on its regular cycle.

Mitsubishi Data Members

As with all Lookout drivers, you can access I/O points and other data through data members. The following is a table of data members currently supported by the Mitsubishi object class.

Table 18-64. Mitsubishi Data Members (A Series)

Data Member	Type	Read	Write	Description
C0–C1023	numeric	yes	yes	Counter, 16-bit word.
CommFail	logical	yes	no	Object-generated signal, ON if Lookout cannot communicate with the device(s).
D0–D1023	numeric	yes	yes	Data register, 16-bit word.
M0–M2047	logical	yes	yes	Discrete coil, 1-bit.
Poll	logical	no	yes	When this expression transitions from FALSE to TRUE, Lookout polls the device.
PollRate	numeric	no	yes	Lookout expression that determines the device polling frequency.
T0–T1023	numeric	yes	yes	Timer, 16-bit word.
Update	logical	yes	no	Object-generated signal that pulses low each time it polls the device.
W0–W1023	numeric	yes	yes	Data register, 16-bit word.
X0–X2047	logical	yes	no	Discrete input, 1-bit.
Y0–Y2047	logical	yes	yes	Discrete output, 1-bit.

Table 18-65. MitsubishiFX Data Members (FX Series)

Data member	Type	Read	Write	Description
C0–C1023	numeric	yes	yes	Counter, 16-bit word.
D0–D1023	numeric	yes	yes	Data register, 16-bit word.
T0–T1023	numeric	yes	yes	Timer, 16-bit word.
X0–X1023	logical	yes	no	Discrete input, 1-bit.

Table 18-65. MitsubishiFX Data Members (FX Series) (Continued)

Data member	Type	Read	Write	Description
CommFail	logical	yes	no	Object-generated signal that is ON if Lookout cannot communicate with the device(s).
Y0–Y1023	logical	yes	yes	Discrete output, 1-bit.
M0–M1023	logical	yes	yes	Discrete coil, 1-bit.
sD8000–sD8255	numeric	yes	yes	Special data register, 16-bit word.
sM0–sM1023	logical	yes	yes	Special discrete coil, 1-bit.
S0–S1023	logical	yes	yes	State, 1-bit.
Poll	logical	no	yes	When this expression transitions from FALSE to TRUE, Lookout polls the device.
PollRate	numeric	no	yes	Lookout expression that determines the device polling frequency.
Update	logical	yes	no	Object-generated signal that pulses low each time it polls the device.

Mitsubishi Status Messages

No response within timeout period
Lookout received no response from a device within the **Receive timeout** period. The Mitsubishi object is able to use the COM port, but when it polls the device, it does not respond—as if it is not even there. If you have daisy-chained several devices, you have introduced an inherent delay. You may need to significantly increase **Receive timeout** (and **PollRate**) to ensure Lookout is allowing enough time to receive the expected response. This increase has nothing to do with the processing capabilities of Lookout, but is based solely on **Data rate** and the number of devices on the chain. Also, verify your Baud rate settings, cable connections, power, configuration settings, COM port settings, and polling addresses.

Incorrect frame check sum (FCS)
The frame was received with an invalid frame check sum. Check for two or more devices with the same address.

Incorrect PC number in response
The frame received had an incorrect source address. Check for two or more devices with the same address.

Incorrect command in response
The frame received had an incorrect command. Check for two or more devices with the same address.

Garbled or Invalid frame
The frame was received without proper termination character (ETX). Check the Lookout **Receive Gap** setting.

No acknowledgment for write frame
The write frame was not acknowledged by the PC. Check the address of Mitsubishi object or the address range of the PC.

Mitsubishi errors reported in the response
These errors are reported by the Mitsubishi device, and are in turn reported to the user in text form.

Mitsubishi Models Supported

A series:

A0J2, A0J2H, A1, A1S, A1N, A1E, A2, A2n, A2C, A2E, A3, A3N, A3E, A3H, A3M.

FX series:

All models through March, 1997.

Modbus
ModbusMOSCAD

Modbus and ModbusMOSCAD are protocol driver classes Lookout uses to communicate with equipment such as programmable logic controllers (PLCs), remote terminal units (RTUs), or any other piece of equipment using Modbus Serial (ASCII or RTU) or Modbus Plus communication protocol.

The Modbus object class has general-purpose addresses, such as holding register 40001, and is suitable for communicating with nearly all Modbus devices, including the Control Microsystems TeleSAFE RTU.

The ModbusMOSCAD object class works with Motorola MOSCAD PLCs and RTUs. It also uses the Modbus Serial (ASCII or RTU) or Modbus Plus communication protocol, but its data members reflect the address of Motorola MOSCAD devices.

You can limit the number of channels Lookout uses on the SA-85 card `.INI` file, as shown in the following example.

```
[ALL]

MaxChannels=channel
```

where *channel* is a number between 1 and 8, inclusive (default = 8).

These protocol driver objects contain a great deal of data. All readable and writable members (inputs/outputs), polling instructions, read/write blocking, serial port usage, and so on are bundled with the object. Therefore, as soon as you create a Modbus or ModbusMOSCAD object you immediately have access to all the object data members (see data member list below).

☞ **Note** ***Lookout protocol driver objects automatically generate an efficient read/write blocking scheme based on the inputs and outputs being used in your process file. You are not required to build your own I/O blocking table.***

As protocol drivers, both object classes conform to the specifications in the Modicon Modbus Protocol Reference Guide PI-MBUS-300 Rev. C. The drivers support ASCII and RTU transmission modes, as well as Modbus Plus.

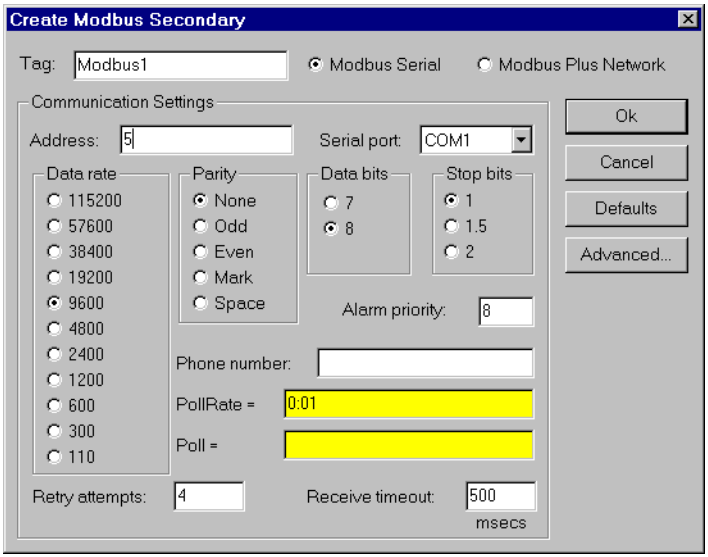


Figure 18-66. Modbus Configuration Parameters Dialog Box

In this example, Lookout is connected to a Modbus-speaking PLC with an address of 5 using serial port 1 (which was previously configured for hardwired communications), and polling the device every second.

Modbus Serial indicates that the slave device talks either Modbus ASCII or Modbus RTU. When you select this option, Lookout first tries to communicate using the RTU format. If unsuccessful, it then tries the ASCII format (a little slower). If your network is susceptible to repeated communication problems, and if these problems slow scanning considerably, you may want to disable Lookout from retrying both formats. This can speed communication retries by Lookout; however, it will not fix your communication problems. Call National Instruments Technical Support to for information on how to prohibit Lookout from trying to communicate using both formats.

Modbus Plus Network indicates that the slave device is connected to the Lookout computer via a Modbus Plus network card.

☞ **Note** ***NetBIOS-based networking software typically uses software interrupt 5C. This is also the default software interrupt used by the Modbus Plus Network card driver. If you are unable to get Modbus Plus and Windows for Workgroups to work at the same time, there is probably a software interrupt conflict. Change the Modbus Plus software interrupt from 5C to 5D, 5E, or 5F. Refer to your Modicon documentation for instructions on changing the software interrupt setting.***

If you select **Modbus Serial**, you must specify **Address**, **Serial Port**, **Data Rate**, **Parity**, **Data Bits**, and **Stop Bits**. And if you are using a Dial-up modem connected to your communication port, you must also specify a **Phone Number**.

If you select **Modbus Plus Network**, you need only specify the remote device **Address**.

Address is a slave address and refers to the PLC or RTU address setting as set on the device dip switches. If devices share a common line, they require unique addresses (1 to 255).

Serial port specifies which port the object uses for communication to the external device. This does not specify the communication type. Communication type is determined by the **Options»Serial Ports…** command.

Data rate, **Parity**, **Data bits**, and **Stop bits** reference the settings on the hardware device.

The **Defaults** button replaces the current settings with default values.

Alarm priority determines the priority level of Modbus-generated alarms.

Phone number specifies the number to be dialed if the serial port setting is configured for dial-up. This number only applies to the individual protocol object.

PollRate is a numeric expression that determines how often to poll the device. Modbus then polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *Introduction*, for information on entering time constants.

Poll is a logical expression. When this expression changes from FALSE to TRUE, Lookout polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Retry attempts specifies the consecutive number of times Lookout attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the Modbus object generates an alarm and releases the communication port back to the communications subsystem which then moves on to the next device in the polling queue (if any). Refer to the communications chapter for more information.

Receive timeout is the time delay Lookout uses in waiting for a response from a device before retrying the request.

Advanced Modbus Parameters

The Modbus driver attempts to block the reads and writes of coils, input registers and holding registers into groups to maximize communication efficiency. Through the **Advanced Modbus Options** dialog box, you can control the maximum block sizes that the driver uses. In fact, if your device does not support the default block sizes, you may have to specify smaller blocks.

The **Advanced…** button invokes the **Advanced Modbus Options** dialog box you can use to customize specific options within the Modbus protocol.

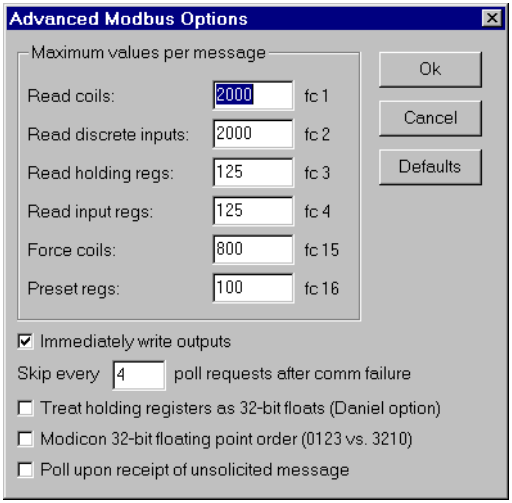

Advanced Modbus Options
Maximum values per message
Read coils: 2000 fc 1
Read discrete inputs: 2000 fc 2
Read holding regs: 125 fc 3
Read input regs: 125 fc 4
Force coils: 800 fc 15
Preset regs: 100 fc 16
Ok
Cancel
Defaults
Immediately write outputs
Skip every 4 poll requests after comm failure
Treat holding registers as 32-bit floats (Daniel option)
Modicon 32-bit floating point order (0123 vs. 3210)
Poll upon receipt of unsolicited message

Figure 18-67. Advanced Modbus Parameters Dialog Box

The Modbus object class uses Modbus Function Codes 01, 02, 03, 04, 05, 06, 15, and 16; and expects the remote I/O device to support these codes as specified by Modbus. The driver can communicate with up to 247 Modbus slave devices on each serial port.

The **Maximum values per message** settings specify the maximum number of elements Lookout attempts to read (`fc 1` – `fc 4`), or write (`fc 15` and `fc 16`), in a single Modbus message. The default values represent the maximum number of elements that the protocol can transmit in a single message, and provides optimal speed. However, some devices are not

capable of handling the maximum number of elements, so you should set the values according to the documentation for those devices.

If the **Immediately write outputs** option is ON, Lookout immediately polls the device any time a value changes that is being written out to the device. If it is OFF, Lookout waits until the next scheduled poll to write out changed values.

The **Skip every…** setting instructs Lookout to not poll a device it has lost communication with on every scheduled poll. Instead, Lookout skips the device in the polling cycle accordingly. Once communications have been reestablished, the device is polled on its regular cycle.

The **Daniel option** is device-dependent and instructs Lookout to treat holding registers as 32-bit IEEE floating values instead of 16-bit values. If you set this flag, you must also set your hardware device to treat holding registers as 32-bit floats—most devices do not support this option, but Bristol-Babcock RTUs and Daniel flow meters do.

☞ **Note** ***Activating the*** *Daniel option* ***deactivates all Modbus holding register members (on that device) except for 40001 – 49999 and 4000001 – 465000. If you attempt to read D40001, for example, the returned value is 0, and Lookout will not attempt to write D40001 to the RTU. Of course, in devices that do not support this option, you can still read and write two adjacent holding registers as a floating point value with the Modbus data members F40001 – F4999. In fact, this is a more general purpose solution than the Daniel option, because you can still read bits and word values out of the holding registers, too.***

Modicon 32-bit floating point order chooses whether the floating point registers are stored in an alternate Modicon format, as required by some Modicon Quantum PLCs.

Modbus Protocol Statistics

The driver monitors Modbus Protocol Statistics. This data is held within readable data members of the Modbus object and you can see them in the Modbus Protocol Statistics dialog box. To view the dialog box, select **Options»Modbus…** and click on **Statistics…**.

Note ***The*** Options»Modbus… ***option is only visible in the Options menu if a Modbus object was previously created in your Lookout application.***

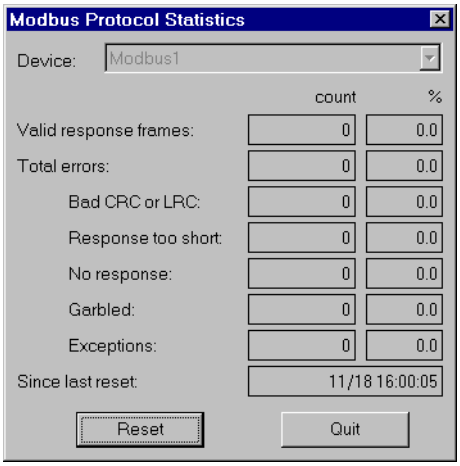


Figure 18-68. Modbus Protocol Statistics Dialog Box

The **Count** column contains the accumulated number of messages received from the selected **Device** that fall into each respective category since the last time the Reset button was pressed. The percent column (**%**) indicates the percentage of messages received that fall into each respective category since the last time the Reset button was pressed.

When you depress the **Reset** button, the **ResetCounts** data member is set TRUE, setting all statistical values to zero. Lookout records the date and time that the reset was last performed in the **Since last reset** data field.

Modbus Data Members

The Modbus object class supports both 5-digit and 6-digit addressing. When you use a 6 digit address, the left-most digit represents the address *type* as follows:

Table 18-66. 6-Digit Address Coding

First Digit	Address Type
0	Single-bit coils
1	Discrete inputs

Table 18-66. 6-Digit Address Coding (Continued)

3	Input registers
4	Holding registers

The remaining 5 digits represent the actual address of the coil, input or holding register.

☞ **Note** ***When you reference address 000001 and address 1, you are referring to the same point, but 40001 and 040001 do not refer to the same point. Because zero is the left-most digit in the 6-digit address, 040001 points to the 40001st single-bit coil and 40001 refers to the first holding register.***

Table 18-67. Modbus Data Members

Data Member	Type	Read	Write	Description
000001 – 065000	logical	yes	yes	6-digit addresses of single-bit coils
1 – 9999	logical	yes	yes	Single-bit coils
10001 – 19999	logical	yes	no	Single bit discrete inputs
100001 – 165000	logical	yes	no	6-digit addresses of single-bit discrete inputs
30001 – 39999	numeric	yes	no	16-bit input registers encoded as unsigned binary integers ranging from 0 to 65535
300001 – 365000	numeric	yes	no	6-digit addresses of 16-bit input registers encoded as unsigned binary integers ranging from 0 to 65535
40001 – 49999	numeric	yes	yes	16-bit holding registers encoded as unsigned binary integers ranging from 0 to 65535
400001 – 465000	numeric	yes	yes	16-bit holding registers encoded as unsigned binary integers ranging from 0 to 65535
40001.1 – 49999.16	logical	yes	yes	Access individual bits out of holding registers and read them as logical ON/OFF values. The least significant bit is 1; the most significant, 16.

Table 18-67. Modbus Data Members (Continued)

Data Member	Type	Read	Write	Description
400001.1 – 465000.16	logical	yes	yes	6-digit address used to access individual bits out of holding registers and read them as logical ON/OFF values. The least significant bit is 1; the most significant, 16.
BadCRC	numeric	yes	no	Number of responses from device whose message failed the cyclic redundancy check (CRC) or the longitudinal redundancy check (LRC)
BCD30001 – BCD39999	numeric	yes	no	16-bit input registers encoded as binary-coded decimal integers ranging from 0 to 9999
BCD300001 – BCD365000	numeric	yes	no	6-digit addresses of 16-bit input registers encoded as binary-coded decimal integers ranging from 0 to 9999
BCD40001 – BCD49999	numeric	yes	yes	16-bit holding registers encoded as binary-coded decimal integers ranging from 0 to 9999
BCD400001 – BCD465000	numeric	yes	yes	6-digit addresses of 16-bit holding registers encoded as binary-coded decimal integers ranging from 0 to 9999
CommFail	logical	yes	no	Driver-generated signal that is ON if Lookout cannot communicate with the device for whatever reason
D400001 – D465000	numeric	yes	yes	6-digit addresses of 32-bit unsigned holding register—reads two adjacent holding registers as a single 32-bit number ranging from 0 to 4,294,967,296.
D40001 – D49999	numeric	yes	yes	32-bit unsigned holding register—reads two adjacent holding registers as a single 32-bit number ranging from 0 to 4,294,967,296.

Table 18-67. Modbus Data Members (Continued)

Data Member	Type	Read	Write	Description
Exceptions	numeric	yes	no	Number of responses from device whose message was understandable to the driver but included an error code indication from the device
F40001 – F49999	numeric	yes	yes	32-bit IEEE floating point register—reads two adjacent holding registers as a single 32-bit floating point value
F400001 – F465000	numeric	yes	yes	6-digit addresses of 32-bit IEEE floating point register—reads two adjacent holding registers as a single 32-bit floating point value
Garbled	numeric	yes	no	Number of responses from device whose message was unintelligible to the driver
NoResponse	numeric	yes	no	Number of polls generated by driver not responded to by device
OffHook	logical	no	yes	When TRUE, this flag instructs the Modbus object to retain exclusive use of its assigned communication port
Poll	logical	no	yes	When this expression transitions from FALSE to TRUE, Lookout polls the device.
PollRate	numeric	no	yes	Lookout expression that determines the device polling frequency.
ProtocolErrors	numeric	yes	no	Total number of bad messages received from polled device
ResetCounts	logical	no	yes	Resets number to zero in the following data members: ValidFrame, NoResponse, TooShort, BadCRC, Garbled, Exceptions, & ProtocolErrors

Table 18-67. Modbus Data Members (Continued)

Data Member	Type	Read	Write	Description
S40001 – S49999	numeric	yes	yes	16-bit holding registers encoded as signed binary integers ranging from –32767 to +32768.
S400001 – S465000	numeric	yes	yes	6-digit addresses of 16-bit holding registers encoded as signed binary integers ranging from –32767 to +32768.
TooShort	numeric	yes	no	Number of responses from device whose message length was too short
Update	logical	yes	no	Driver-generated signal that pulses each time the driver polls the device
ValidFrame	numeric	yes	no	Number of good messages received from polled device

Comments ***You can use the OffHook data member to enhance communications when using the Modbus object class with dial-up modems. When OffHook is TRUE and the serial port is connected to a dial-up modem, the Modbus object does not hang up the modem when the poll is complete. Rather, it keeps the phone off the hook, retaining exclusive use of the serial port. As long as OffHook is TRUE, the Modbus object continues to poll the same PLC without hanging up the modem.***

As soon as OffHook goes FALSE, the object releases the serial port to the communications subsystem, which goes to the next poll request in the queue, if any. The object also releases the port if data communications are lost for any reason—such as if the PLC modem breaks the connection.

When using OffHook, consider defining the driver object **PollRate** to poll fast when OffHook is TRUE, and poll at its normal rate when OffHook is FALSE. You might tie a Switch object to the OffHook writable data member for this very purpose.

ModbusMOSCAD Data Members

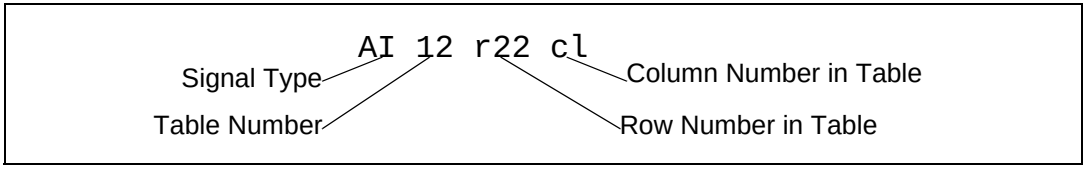
Table 18-68. ModbusMOSCAD Data Members

Data Member	Type	Read	Write	Description
AI0r0c0 – AI30r255c7	numeric	yes	no	Each address represents a 16-bit analog input encoded as an unsigned integer ranging from 0 to 65535
AO0r0c0 – AO30r255c7	numeric	yes	yes	Each address represents a 16-bit analog output encoded as an unsigned integer ranging from 0 to 65535
DI0r0c0.0 – DI30r255c7.15	logical	yes	no	Each address represents an individual discrete input read as a logical ON/OFF value. The least significant bit is 0; the most significant, 15.
DO0r0c0.0 – DO30r255c7.15	logical	no	yes	Each address represents an individual discrete input read as a logical ON/OFF value. The least significant bit is 0; the most significant, 15.
BadCRC	numeric	yes	no	Number of responses from device whose message failed the cyclic redundancy check (CRC) or the longitudinal redundancy check (LRC)
CommFail	logical	yes	no	Driver-generated signal that is ON if Lookout cannot communicate with the device for whatever reason
Exceptions	numeric	yes	no	Number of responses from device whose message was understandable to the driver but included an error code indication from the device
Garbled	numeric	yes	no	Number of responses from device whose message was unintelligible to the driver
NoResponse	numeric	yes	no	Number of polls generated by driver not responded to by device
Poll	logical	no	yes	When this expression transitions from FALSE to TRUE, Lookout polls the device.

Table 18-68. ModbusMOSCAD Data Members (Continued)

Data Member	Type	Read	Write	Description
PollRate	numeric	no	yes	Lookout expression that determines the device polling frequency.
ProtocolErrors	numeric	yes	no	Total number of bad messages received from polled device
ResetCounts	logical	no	yes	Resets number to zero in the following data members: ValidFrame, NoResponse, TooShort, BadCRC, Garbled, Exceptions, & ProtocolErrors
TooShort	numeric	yes	no	Number of responses from device whose message length was too short
Update	logical	yes	no	Driver-generated signal that pulses each time the driver polls the device
ValidFrame	numeric	yes	no	Number of good messages received from polled device

Comments ***To specify the address of an analog input located in row 22, column 1 of table 18-12, you would enter AI12r22c1, where***



ModbusSlave

ModbusSlave is an object class Lookout uses to act as a slave to equipment such as PLCs, distributed control systems (DCSs), or any other device that can act as a Modbus master. The ModbusSlave object enables Lookout to respond to unsolicited messages generated by other devices on a Modbus protocol network.

Applications include:

- PLCs and/or RTUs dial up a Lookout application over a non-dedicated phone line when there is an alarm or event that must be reported (unsolicited report-by-exception).
- Two Lookout packages communicate with each other over radio. One uses a Modbus object and the other uses a ModbusSlave object.
- Lookout, directly connected to a Modbus network, responds to polls from a DCS as a Modbus slave.

Think of the ModbusSlave object as representing a virtual PLC within Lookout. It can receive reads and writes over a Modbus protocol-based network. Like the Modbus object, the ModbusSlave object has a collection of readable and writable data members, all created when you define the ModbusSlave object.

You can create a unique ModbusSlave object for every master PLC, or you can create a single ModbusSlave object for multiple master PLCs. In the latter case, one master might write to one group of registers within the ModbusSlave object and another master might write to another group of registers within the ModbusSlave object.

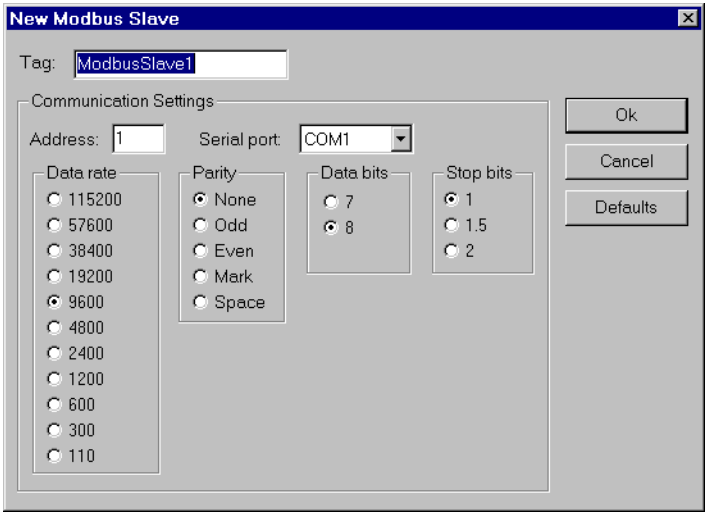


Figure 18-69. Modbus Slave Configuration Parameters Dialog Box

Address is a number (1 to 255) that refers to the address that this ModbusSlave object will assume. Even though this object exists only in software, other physical devices do not know that. They communicate with this object using a physical communication line and need to know the slave address.

Serial port identifies which communication port on the PC that the Modbus master uses to communicate with the ModbusSlave in Lookout.

Data rate, **Parity**, **Data bits**, and **Stop bits** reference the settings utilized by the master hardware device.

The **Defaults** button replaces the current settings with default values.

Modbus Slave Data Members

Table 18-69. ModbusSlave Data Members

Data Member	Type	Read	Write	Description
1 – 9999	logical	yes	yes	Single-bit coils
10001 – 19999	logical	yes	yes	Single bit discrete inputs

Table 18-69. ModbusSlave Data Members (Continued)

Data Member	Type	Read	Write	Description
30001 – 39999	numeric	yes	yes	16-bit input registers encoded as unsigned binary integers ranging from 0 to 65535
40001 – 49999	numeric	yes	yes	16-bit holding registers encoded as unsigned binary integers ranging from 0 to 65535
BCD30001 – BCD39999	numeric	yes	yes	16-bit input registers encoded as binary-coded decimal integers ranging from 0 to 9999
BCD40001 – BCD49999	numeric	yes	yes	16-bit holding registers encoded as binary-coded decimal integers ranging from 0 to 9999
D30001 – D39999	numeric	yes	yes	32-bit input register
D40001 – D49999	numeric	yes	yes	32-bit unsigned holding register—reads two adjacent holding registers as a single 32-bit number ranging from 0 to 4,294,967,296.
F30001 – F39999	numeric	yes	yes	32-bit input register
F40001 – F49999	numeric	yes	yes	32-bit IEEE floating point register—reads two adjacent holding registers as a single 32-bit floating point value
H40001.f1 – H49999.f16	logical	yes	yes	Access individual bits out of holding registers and read them as logical ON/OFF values. The least significant bit is F1 and the most significant bit is F16.
S40001 – S49999	numeric	yes	yes	16-bit holding registers encoded as signed binary integers ranging from –32767 to +32768.

Multistate

Multistate displays different graphics on a control panel as dictated by the values of **Conditional expressions**. You can use up to six **Graphic files**, but at least one is required. Multistate determines which graphic to display based on the order and current status of your **Conditional expressions**. If several **Conditional expressions** are true at once, Multistate displays the graphic associated with the first true expression.

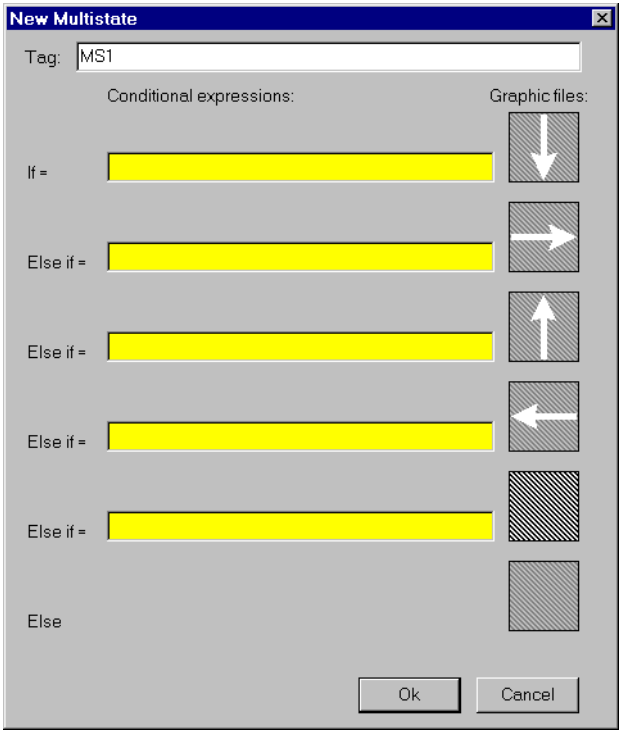


Figure 18-70. Multistate Configuration Parameters Dialog Box

Conditional expressions must result in logical values (TRUE or FALSE). See the *Animator* section for more information about constructing logical statements.

Multistate Data Members

Table 18-70. Multistate Data Members

Data members	Type	Read	Write	Description
none	—	—	—	Multistate objects do not have data members

Comments ***By creating several graphic images that depict a sequence of events, Multistate can be used to create animation sequences on control panels such as hydraulic pistons moving back and forth. A more typical use of Multistate is for three-color pilot lights, where green represents running, red represents stopped, and yellow represents failed, for example.***

For smooth, high speed animations, use the Animator object.

Related Objects ***Animator***

National Instruments Fieldbus

Lookout can communicate with Fieldbus devices using the National Instruments NI-FBUS AT card.

The Lookout Fieldbus objects contain no inherent physical data members. All physical data members are obtained by reading Device Descriptions (DD). A default DD is provided when you install the National Instruments Fieldbus card which contains all the standard Fieldbus blocks. Lookout creates a Fieldbus object class dynamically for every uniquely named block found in any of the DDs. If a block is defined more than once by name, then the user is able select which DD block definition to apply to the Lookout object in the object creation dialog box. The National Instruments supplied DD is selected by default.

In order to add or remove a Fieldbus block type to the Lookout object class list, delete the `Lookout.dat` file in the `\Lookout` directory and restart Lookout. This forces Lookout to re-index all of its subordinate object classes. The name of Fieldbus object classes is the name of the block preceded by `FF_`. For example, the Lookout class name for an Analog Input block would be `FF_Analog_Input`.

Lookout also supports manufacturer supplied DDs. These DDs must be placed in the appropriate directory structure as defined by the Fieldbus Foundation. You can configure the base directory for manufacturer supplied DDs with the National Instruments `NI-FBUS Config` software. After selecting **DD Info** in the main dialog box of the `NI-FBUS Config` software, enter the path for your base directory. The DD should be placed, along with its symbol file, two folders below this base directory. Conventions for naming these directories are outlined by Fieldbus Foundation but the names are not used by Lookout.

When re-indexing occurs, Lookout looks into every DD in the directory system for Fieldbus blocks. Lookout also requires a symbol file (`*.sym`) for every DD (`*.ffo`).

☞ **Note** ***DD must be made with tokenizer version 4.2 or greater.***

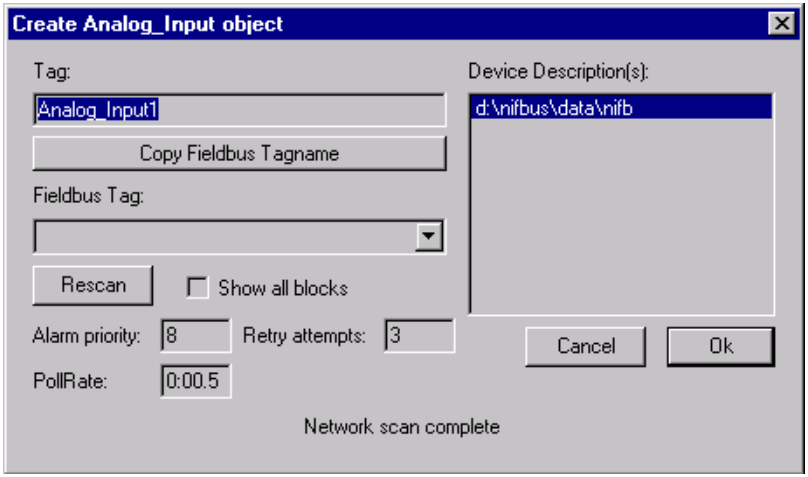


Figure 18-71. Fieldbus Configuration Parameters Dialog Box

Tag is the Lookout tag name and is used to reference this object in Lookout.

Copy Fieldbus Tagname replaces invalid characters in the Fieldbus Tagname with underscores while copying the Fieldbus Tag into the Lookout Tag edit box.

Fieldbus Tag shows a list of all Fieldbus blocks, of the appropriate type, available on your network. This list comes from comparing the DD_Item_Type of the block on the networks with the type of object that the user has selected to create. If there are no devices in this listbox, you can click the **Show all blocks** checkbox and select the **Rescan** button to show Fieldbus blocks of all types on the network.

☞ **Note** ***Lookout treats the block selected as the type of block the user was trying to make. If an incorrect block is selected, the wrong parameter list appears for that block and invalid data request may occur.***

Device Description contains a list of Device Descriptions that have definitions for the block the user is about create. The parameter list for each definition may vary. Care should be taken in selecting the proper DD for the block this Lookout object will represent.

PollRate is a numeric expression that determines how often to poll the device. The object then polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See

Numeric Data Members in Chapter 2, *Introduction*, for information on entering time constants.

Retry is the number of Fieldbus communication alarms to ignore before reporting an alarm.

Alarm priority determines the priority level of alarms generated by the National Instruments Fieldbus object. Such alarms are typically related to communications with the physical device.

Fieldbus Alarms

The Fieldbus Alarms dialog box lets you acknowledge Fieldbus Alarms.

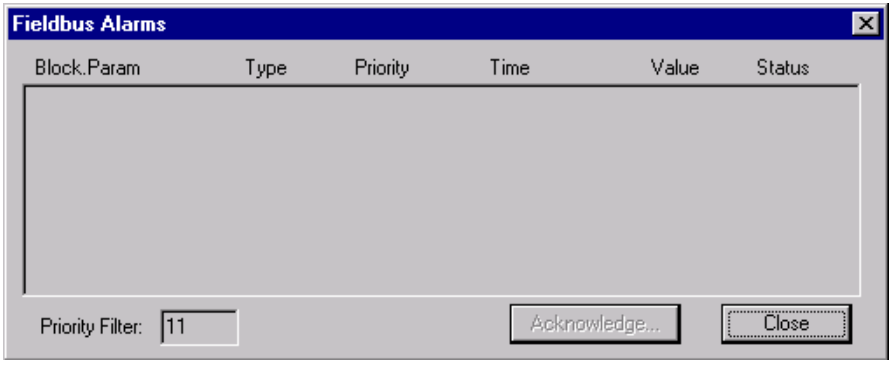


Figure 18-72. Fieldbus Alarms Dialog Box

Priority Filter filters alarms by priority. If you **Acknowledge** an alarm, it is removed from the list if successful.

Fieldbus Data Members

As with all Lookout drivers, you can access I/O points and other data through data members. The following is a table of built-in data members currently supported by the National Instruments Fieldbus object class. All other data members are obtained from device descriptions.

Table 18-71. National Instruments Fieldbus Data Members

Data Member	Type	Read	Write	Description
Alarms	numeric	yes	no	Expression that shows number of active Fieldbus alarms.
Poll	logical	no	yes	When this value transitions from FALSE to TRUE, Lookout polls the device.
CommFail	logical	yes	no	Object-generated signal that is on if Lookout cannot communicate with the device(s).
Update	logical	yes	no	Object-generated signal that pulses low each time it polls the device.
ShowAlarms	logical	no	yes	When this value transitions from FALSE to TRUE, Lookout shows the Fieldbus alarm window.

Fieldbus Status Messages

No response within timeout period
Lookout did not receive a response from a device within the **Receive timeout** period. Check the devices connection. Check to see if the device can be polled using NI-FBUS Dialog.

Cannot find nifb.dll
Lookout could not find a component needed to communicate with the Fieldbus devices. This component is placed during the installation of the NI-FBUS software. Make sure the installation was successful.

Could not open NIFB session [: optional error message]
Lookout could not open a Fieldbus communications session. If available, specific error text is shown providing specific information. Make sure that the `nifb` driver in **Settings»Control Panels»Devices** is started. Also make sure that `nifb.exe` is running.

Block tag could not be found on network
Lookout could not find the block tag specified on the Fieldbus network. Check tag spelling or the device/network connections.

I/O error: [error message]
Lookout encountered an error during the read/write process. The error message, provided by the Fieldbus communication interface, contains specific information regarding the cause of the failure.

Fieldbus Troubleshooting

If after deleting the lookout.dat file and restarting Lookout there are no FF_(Fieldbus Object) classes listed in Lookout; check the following:

- Is the path to the data directory in your `nifbus` installation directory specified correctly in NI-FBUS config software under **DD Info**?
- Is the path to the base directory for manufacturer supplied DDs specified correctly?
- Does the `NI-FBUS Dialog` application work correctly?

Lookout cannot establish a fieldbus session with the card.

- Is `NIFB.EXE` running?
- Has the device driver `nifb` in your device control panel been started?
- Does the `NI-FBUS Dialog` application work correctly?

National Instruments FieldPoint

FieldPoint is a protocol driver class Lookout uses to communicate with FieldPoint devices using an enhanced version of the Optomux communication protocol. This object works with the FieldPoint models FP-1000, FP-1001, FP-AI-110, FP-AO-200, FP-DI-330, and FP-DO-400

This protocol uses no parity, eight data bits and one stop bit. In Lookout, a single FieldPoint object represents all devices connected to the same COM port.

The Lookout FieldPoint object can read and write to all predefined data points allowed by the particular FieldPoint module. When you create a FieldPoint object, you have immediate access to all the object data members. See the *Data Members* section for more information on object data members.

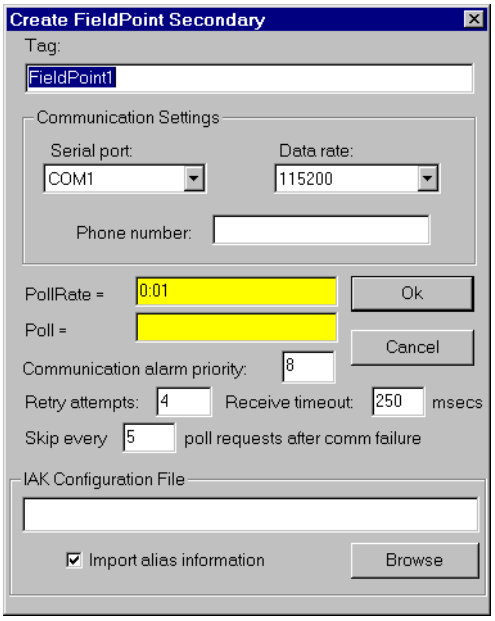


Figure 18-73. National Instruments FieldPoint Configuration Parameters Dialog Box

Serial port specifies which COM port the object uses for communicating to the external device. This does not specify the communication type. Communication type is determined by the **Options»Serial Ports...** menu command.

Data rate indicates the baud rate that Lookout uses to communicate with the hardware device. This setting should match the selection made on each of your network modules.

Phone number specifies the number to be dialed if the selected serial port is configured for dial-up. This number only applies to the individual protocol object.

PollRate is a numeric expression that determines how often to poll the device. The object then polls the device at the specified time interval. Ordinarily, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *Introduction,* for information on entering time constants.

Poll is a logical expression. When transitioned from FALSE to TRUE, Lookout polls the device. This can be a simple expression, like the signal from a pushbutton, or it can be a complex algorithm.

Communication alarm priority determines the priority level of alarms generated by the FieldPoint object. Such alarms are typically related to communications with the physical device.

Retry attempts specifies the consecutive number of times Lookout attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the FieldPoint object generates an alarm and releases the COM port. Refer to Chapter 6, *Serial Communications,* for more information.

Receive timeout is the time delay Lookout uses in waiting for a response from a device before retrying the request.

The **Skip every n** setting instructs Lookout to not poll a device it has lost communication with on every scheduled poll. Instead, Lookout skips the device in the polling cycle. Once communications have been reestablished, the device is polled on its regular cycle.

IAK configuration file is a dialog for selecting an IAK configuration file. The IAK file contains alias and scaling information which is extracted for use in Lookout. Choose the configuration file you want to use by entering the path directly, or use the **Browse** button. Check **Import alias information** if you want to extract information from the selected file.

FieldPoint Data Members

As with all Lookout drivers, you can access I/O points and other data through data members. The following is a table of data members currently supported by the FieldPoint object class.

Table 18-72. FieldPoint Data Members

Data Member	Type	Read	Write	Description
AI000.00 - AI255.15	numeric	yes	no	Analog input channels
AO000.00 - AO255.15	numeric	yes	yes	Analog output channels
DI000.00 - DI255.15	logical	yes	no	Discrete input channels
DO000.00 - DO255.15	logical	yes	yes	Discrete output channels
Update	logical	yes	no	Goes high when Lookout begins a poll cycle on the device
CommFail	logical	yes	no	Goes high if Lookout cannot communicate with the device
Poll	logical	no	yes	When transitioned from low to high, Lookout begins a poll cycle on the device
PollRate	numeric	no	yes	Specifies the frequency at which Lookout polls the device

☞ **Note** ***The first two characters of the I/O data members represent the kind of module being accessed. The next three digits represent the device address of the module. This is the address of the I/O module itself, not the network module that governs it. Following the period are two digits representing the channel number within the module.***

Not all of these data members are valid for every FieldPoint module. For all the device types you are able to select the full range of device addresses and channels. So if you select DO123.03, you need to be certain that the device at address 123 is in fact a discrete output module.

For a more complete definition of the function of these data members, see FieldPoint documentation.

☞ **Note** ***In the event of a power cycle to the FieldPoint device during use, the configuration of the device reverts to some default state, which is configurable. You should keep in mind that if the ranges you configured into the IAK file differ from those in the***

power-up configuration, the scaling information imported from the IAK file and used as a Lookout alias might become outdated and incorrect after a power loss. To avoid this, make certain your power-up configuration ranges and your IAK configuration ranges are identical.

FieldPoint Multiple Discrete Data Members

Table 18-73. Multiple Discrete Data Members

Data Member	Type	Read	Write	Description
MDI000.0000 - MDI255.FFFF	numeric	yes	no	Multiple discrete input channels
MDO000.0000 - MDO255.FFFF	numeric	yes	yes	Multiple discrete output channels

These special purpose data members are for reading or writing a numeric integer value to a set of discrete channels.

For instance, when you are configuring your modules with FieldPoint Explorer, you have the option of selecting more than one discrete channel for a data item that you are defining. If you do this and import the resulting `.IAK` file into Lookout for use as aliases, the aliases created will correspond to this set of data members. You can then read and write to all the discrete channels with a single numeric data member. The data member names are in the form `MTTAAA.CCCC`, where:

`M` Indicates multiple as opposed to single

`TT` Two characters specifying module type (Discrete Out, Discrete In)

`AAA` Three numeric characters specifying module address

`CCCC` Four hexadecimal characters specifying which of the 16 channels are included in this data member

☞ **Note** ***These data members will not enumerate. You may use them either by importing configurations from FieldPoint Explorer or by entering the data member name explicitly.***

FieldPoint Error Messages

No response within timeout period
Lookout received no response from a device within the **Receive timeout** period. The FieldPoint object is able to use the COM port, but when it polls the device, the device does not respond. If you have daisy-chained several devices, you have introduced an inherent delay. You may have to significantly increase **Receive timeout** (and **Poll Rate**) to ensure Lookout is allowing enough time to receive the expected response. This increase has nothing to do with the processing capabilities of Lookout. Rather it is based solely on **Data rate** and the number of devices on the chain. Also, verify your baud rate settings, cable connections, power, configuration settings, COM port settings, and polling addresses.

Module returning ?? checksum
This means that the frame sent from the PLC in response to the command sent by Lookout out returned `??` instead of a valid checksum. Check FieldPoint configuration.

Message Garbled - Bad CRC
This means the checksum (`CRC` in this case) failed in a frame received by Lookout. Check cabling or for two or more devices with the same address.

Unexpected data response length
The frame received was of an unexpected length. Check the Lookout **receive gap** setting.

Error loading IAK configuration file
Lookout was not able to successfully extract data from the `.IAK` configuration file. Try running the FieldPoint Explorer again and reconfigure your hardware.

FP error: Power-up clear expected
A command other than power-up clear was attempted after power-up or power failure. The command sent is ignored and normal operations should resume.

FP error: Undefined command
The addressed module does not support this command. (for example, trying to write to an input module) Check to see if you are sending a command appropriate to the module.

FP error: Checksum error
This means the checksum (`CRC` in this case) failed in a frame sent by Lookout. Check the Lookout **receive gap** setting.

FP error: Input buffer overrun
The command sent to the FieldPoint module was too long. Check the Lookout **receive gap** setting.

FP error: Non-printable ASCII character received
Only characters from ASCII value 33 to 127 are permitted in FieldPoint commands. The command is ignored.

FP error: Data field error
An insufficient or incorrect number or characters were received by the FieldPoint module for the specified command. Check the Lookout **receive gap** setting.

FP error: Communications link network watchdog timed out
There has been no network traffic in the amount of time specified by your watchdog configuration settings, and the system has reverted to its watchdog defaults.

FP error: Specified limits invalid for the command
This includes the case where an invalid digit (hex or decimal) was received. Check the Lookout **receive gap** setting.

FP error: ASCII to binary conversion error
One or more ASCII characters could not be converted to binary on the FieldPoint module. Check the Lookout **receive gap** setting.

FP error: Invalid device address
The command is valid, but the addressed module does not support the command received. Check to see if you are sending a command appropriate to the module.

FP error: Serial framing error
An improperly framed command was received by the FieldPoint module. Check the Lookout **receive gap** setting.

FP error: Addressed module does not exist
Make sure that you are addressing a valid module address.

FP error: Invalid channel
One or more channels specified in the command either do not exist or do not support the operation specified.

FP error: Invalid range setting
Check to see that the range information on the module has not changed, possibly due to a loss of power.

FP error: Invalid operation for the module
One or more module-specific operations specified in the command either do not exist or do not support the operation specified. Make sure that you are not requesting a discrete operation for an analog module, and vice versa.

FP error: Module has been hotswapped since last command
The alarm should deactivate immediately after it appears. It appearance is only to acknowledge that a hot swap has occurred.

FP error: Irrecoverable hardware fault
A malfunction in the FieldPoint firmware or hardware has made communications from Lookout impossible.

Channel specific error: dev:##,ch:##,err:##
These are error codes returned from the FieldPoint I/O modules. The alarm message specifies a device address, channel number, and error code. See FieldPoint documentation for a description of the error condition.

National Instruments Lookout OPC Client Driver

The National Instruments Lookout OPC Client Driver is a protocol driver class Lookout uses to read data from and write data to any OPC server. It supports numeric, logical (boolean), and text I/O. It can read from the OPC server cache for maximum speed or, at the your discretion, can read critical data directly from the device. You can also set the update rate of the server's cache.

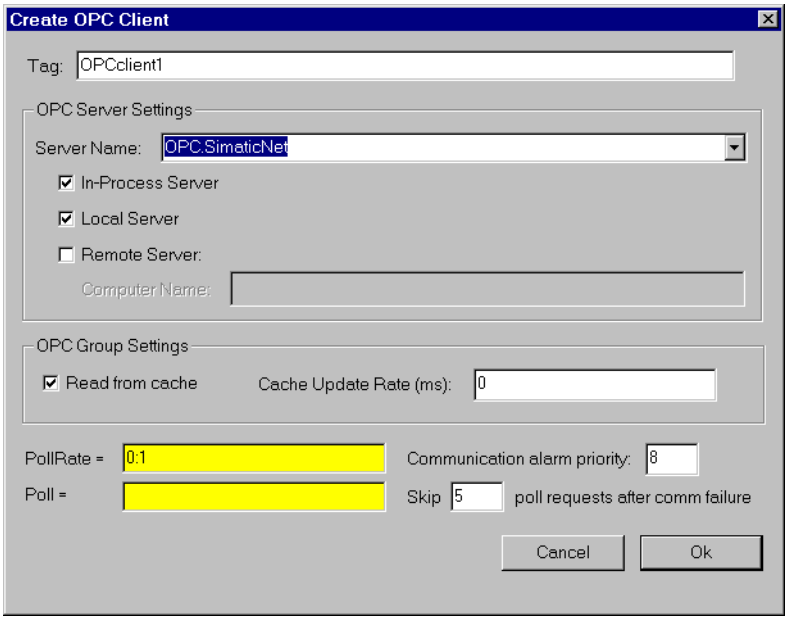


Figure 18-74. OPC Client Configuration Parameters Dialog Box

The **Server Name** box enumerates all of the OPC servers registered on the local computer. Select the appropriate server.

The **In-Process Server**, **Local Server**, and **Remote Server** specify which types of servers the OPC Client will attempt to launch. The OPC Client attempts to launch and connect to the selected servers in the order of **In-Process Server**, **Local Server**, and **Remote Server**. It stops as soon as it successfully connects to a server. If it cannot successfully connect to any of the selected servers, the OPC CLient generates an alarm. Select **Remote Server** to enable **Computer Name**, which specifies the name of the computer on which the remote server is to be launched.

If the **Read from cache** box is checked, the OPC Client object requests data from the server cache. This is the preferred method of requesting data from the server, because the server can respond immediately to a read request. However, the possibility exists that the data in the cache might be stale, because it might not have been updated recently (see **Cache Update Rate** parameter for more information).

If the **Read from cache** box is unchecked, the OPC Client object requests the server to read data directly from the device. This guarantees that the data is current. However, it takes longer for the server to process the request, as well as delaying the servicing of all other clients requests. Reading directly from the device should be reserved for critical data.

Cache Update Rate specifies the time interval in milliseconds at which the server should update the data in the cache *for that particular OPC Client object*. The acceptable range is 0–4000000000. Using 0 requests the server to update the cache as fast as possible.

PollRate is a numeric expression that determines how often the OPC Client object polls the OPC server. The object then polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *Introduction*, for information on entering time constants.

Poll is a logical expression. When this expression changes from FALSE to TRUE, Lookout polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Communication Alarm Priority specifies the priority of alarms generated by this OPC Client object.

Skip XX poll requests after COM failure specifies how many polls to the OPC server to skip after an error occurred trying to communicate with the server. After skipping the requested number of polls, the OPC Client attempts to resume normal polling of the server.

OPC Client Data Member Tables

As with all Lookout drivers, you can access I/O points and other data through data members. The Lookout OPC Client object class currently supports the data members contained in the following table.

Table 18-74. OPC Client Data Members

Data Member	Type	Read	Write	Description
CommFail	logical	yes	no	Goes high if Lookout cannot communicate with the server
DataError	logical	yes	no	Goes high is the object cannot properly process the data returned by the server
N0 - N99999	numeric	yes	yes	Numeric I/O
L0 - L99999	logical	yes	yes	Logical I/O
Poll	logical	no	yes	When transitioned from FALSE to TRUE, the object polls the server
PollRate	numeric	no	yes	Specifies the frequency at which Lookout polls the server
T0 - T99999	text	yes	yes	Text I/O
Update	logical	yes	no	Goes high when Lookout begins a poll cycle on the device; goes low when a poll ends successfully

OPC Item IDs in Lookout

OPC item IDs may be composed of a wider range of characters than Lookout data member names. In addition, there must be a way of specifying an optional access path for an OPC item ID. Therefore, you must create a Lookout alias for each OPC item ID that you want to access. You then use the Lookout alias to access the corresponding OPC item ID.

The OPC Client uses native data member names of N0 - N99999 to identify numeric I/O points, L0 - L99999 to identify logical (boolean) I/O points, and T0 - T99999 to identify text I/O points. To access an I/O point in an OPC server, create an alias for a native data member name of the appropriate type.

Use the Object Database Editor to enter an alias in Lookout (see the *Editing Object Databases* section of Chapter 4, *Using Lookout*, for more detailed information on creating an alias). You may enter aliases individually, or import lists of aliases from Excel 4 spreadsheet.

OPC Item ID Format in Lookout

The OPC Client uses the **Description** field of an alias to describe the OPC item ID. The format of an OPC item ID is:

~OPC access path~OPC item ID

That is, a tilde, followed by the access path, followed by another tilde, followed by the OPC item ID. An access path consisting of an empty string is denoted by 2 consecutive tildes, as shown in the following example.

~~OPC item ID

A Null access path is denoted by a single tilde, as shown in the following example.

~OPC item ID

Examples

`~Modbus Demo Box~4:0`

The access path is `Modbus Demo Box` and the OPC item ID is `4:0`.

`~~4:0`

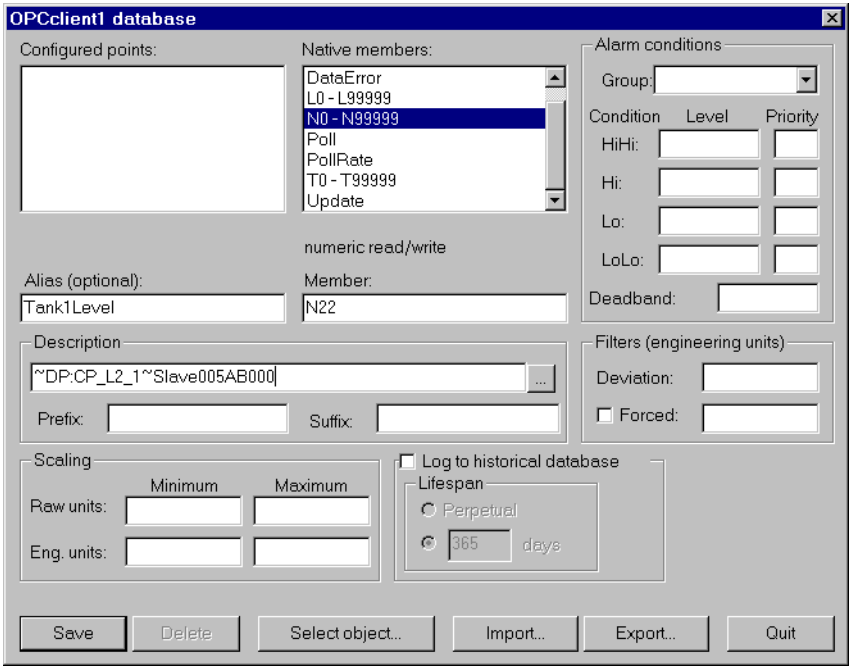
The access path is an empty string and the OPC item ID is `4:0`.

`~4:0`

The access path is Null and the OPC item ID is `4:0`.

Adding a Single Alias

1. Create an OPC Client in Lookout.
2. From the menu, select **Object»Edit Database**. The **Edit object database** dialog pops up.
3. Select the OPC Client object you want to add an alias to.
4. Click **OK** to bring up the **OPC Client database** dialog.
5. Enter a native data member name, such as N22 (N for a numeric I/O point, L for a logical I/O point, T for a text I/O point), in the **Member** edit box.

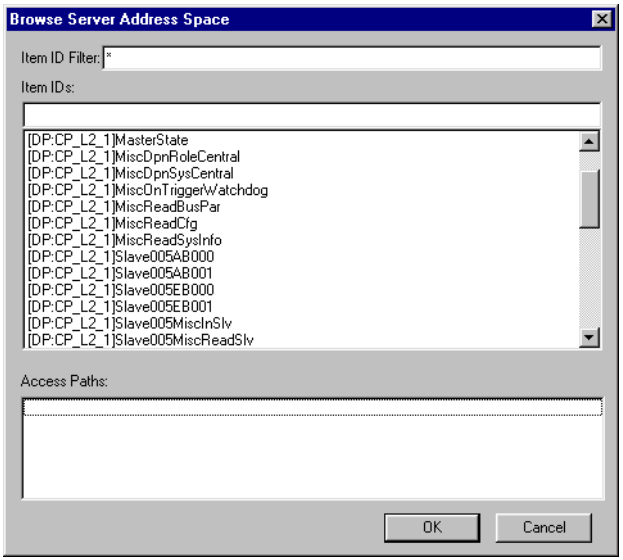


6. Enter a valid Lookout alias, such as `Tank1Level`, in the **Alias** edit box. This is the name that shows up in Lookout when you want to connect to an I/O point.
7. Enter the OPC item ID in the **Description** edit box using the format described in the *OPC Item IDs in Lookout* section. If the OPC server supports browsing, you can click on the button to the right of the Description field to browse the server's address space (see *Browsing OPC Server Address Space* for more information). An error message appears if you try to browse the address space of a server that does not allow browsing.
8. Select **Save** to save the new alias to the database.
9. Select **Quit** when you have entered all your aliases.
10. Once you finish, any time you connect to `Tank1Level` in Lookout you access `N22`, which is the OPC Item ID Slave005AB000and the access path DP:CP_L2_1 as defined in the **Description** field.

Aliases created for an object are specific to that object, so you must create the necessary aliases for each OPC Client object you create.

Browsing OPC Server Address Space

Use the Browse Server Address Space dialog to query the OPC server for a list of names available for use with that particular server.



Use the **Item ID Filter** to display a subset of the available OPC item IDs. Scroll through the **Item IDs** list box to see the available item IDs. When you select an item ID, it is displayed the in the **Item IDs** edit box. Any pertinent access paths for the selected item ID appear in the **Access Paths** list box. You can then modify the item ID in the edit box as needed if the selected item ID is a mnemonic, such as `Slave000-Slave999`. Click the **OK** button to build the fully qualified item ID from the **Item IDs** edit box and the currently selected access path, if any. The ID automatically appears in the **Description** field of the **Edit Database** dialog box.

Importing Alias Lists

You can import alias lists to make alias creation easier.

The simplest way of doing this is to create an alias import file that you can then import for each new OPC Client object. An alias import file is an Excel 4 spreadsheet file. The easiest way to create an importable spreadsheet file is to create a single alias as demonstrated in the *Adding a Single Alias* section, and to export it to an alias import file.

You can then use Excel to enter aliases for all of the required OPC item Ids, following the example you exported. After you have this alias file set up, you can import the file from the Object Database Editor in Lookout any time you need to. Notice that, if you use Excel to modify a Lookout alias import file, you must save the file as an Excel 4 worksheet.

Neutralzone

Neutralzone is an ON/OFF Controller. It functions the way a home air conditioning thermostat does; if the temperature rises above a certain level, the `above` data member goes TRUE (turning the A/C on). When it drops below a lower temperature, the `below` data member goes TRUE (turning the A/C off).

When the incoming **Signal** value rises above both **Low limit** and **High limit**, the data member `above` turns on, and the data member `below` turns off. When the incoming **Signal** value drops below both **Low limit** and **High limit**, `above` turns off, and `below` turns on. The `above` and `below` data members do not change state when the signal value falls back within the two limits (within the neutral zone). **Signal**, **High limit**, and **Low limit** are all numeric expressions.

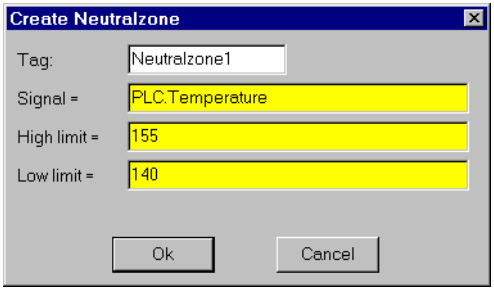


Figure 18-75. Neutralzone Definition Parameters Dialog Box

The previous discussion assumes numeric constants for both limits. However, you could use variable setpoint signals from Pot objects so an operator could dynamically adjust neutralzone behavior.

☞ **Note** ***Neutralzone does not have a display parameters dialog box. You can easily display the result of Neutralzone output signals by referencing its data members in an expression.***

NeutralZone Data Members

Table 18-75. Neutralzone Data Members

Data Members	Type	Read	Write	Description
above	logical	yes	no	ON if signal is greater than both limits, OFF if signal is less than both limits, and does not change if signal is between both limits
below	logical	yes	no	ON if signal is less than both limits, OFF if signal is greater than both limits, and does not change if signal is between both limits

Comments ***You can use this object to turn pumps on and off or open and close valves based on line pressures or tank levels. Neutralzone objects prevent pumps from cycling on and off around a single setpoint, just as an air conditioning thermostat prevents your home air conditioner from incessantly starting and stopping.***

Often the term deadband is mistakenly used to describe a neutral zone. However, deadbands refer to the amount of change a numeric value must travel in the reverse direction before the output numeric value begins to change.

NIDAQDevice

Lookout uses the NIDAQDevice object class to communicate with National Instruments data acquisition devices, including data acquisition devices connected in parallel mode to SCXI hardware. Refer to the NISCXI object class for more information concerning configurations using SCXI hardware connected in multiplexed mode.

To use this object, you should have NI-DAQ 5.0 or better software installed. While the NIDAQDevice object works with some earlier versions of NI-DAQ software, it does not perform well with these earlier versions, and has not been extensively tested.

Consult your NI-DAQ hardware and software manuals for information on installing and configuring National Instruments data acquisition software and hardware.

When you create an NIDAQDevice object for the first time, or when you launch Lookout and run an application using the NIDAQDevice object, Lookout automatically loads the NI-DAQ software. This can take a few moments, during which the screen is frozen. Once the NI-DAQ software has loaded, Lookout returns to its former speed.

☞ **Note** ***This object class is available on 32-bit versions of Lookout 3.7 and later. It is not backward compatible with earlier versions of Lookout, and does not run on the 16-bit version of Lookout.***

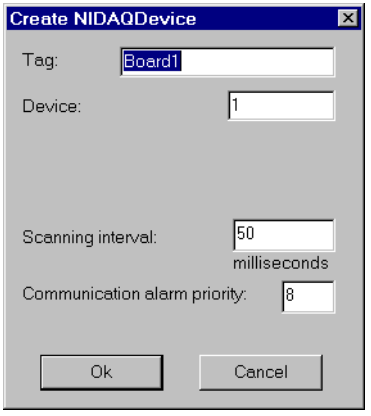


Figure 18-76. NIDAQ Device Configuration Parameters Dialog Box

Device is the NI-DAQ device number. The DAQ configuration utility (`WDAQConf.exe`) assigns this number to an installed device. Valid device numbers range is from 1 to 16.

Scanning interval is the time period between analog and digital input polls. The valid range is 20 msec to 1 day (expressed in msec). You should also keep in mind that some National Instruments DAQ cards need time to stabilize, and so that a scanning rate that is faster than the stabilization rate of the card returns suspect values.

Communication alarm priority determines the priority level of the alarms generated by the NIDAQDevice object.

NIDAQDevice Data Members

Table 18-76. NIDAQDevice Data Members

Data Member	Type	Read	Write	Description
AI0, AI63	numeric	yes	no	Analog input channel in volts.
AO0, AO63	numeric	no	yes	Analog output channel in volts.
CommFail	logical	yes	no	Object-generated signal that is on if, for whatever reason, Lookout cannot communicate with or control the device without error.
DI0, DI31	logical	yes	no	Digital input line.
DO0, DO31	logical	no	yes	Digital output line.
Update	logical	yes	no	Object-generated signal that pulses low each time the object polls the device.

NIDAQ.INI

You may configure individual channel attributes by editing the `NIDAQ.INI` file in your Lookout directory.

To specific a channel, use the following format: `[DEV1.AI5]`

This specifies analog input channel 5 on device number 1.

UPPERLIMIT specifies the upper input limit in volts (for example, `UPPERLIMIT=2.5`). Usually used in conjunction with **LOWERLIMIT**.

LOWERLIMIT specifies the lower input limit in volts (for example, `LOWERLIMIT=-3.7`). Usually used in conjunction with **UPPERLIMIT**.

INPUTMODE specifies the analog input connection mode to use for this channel. Valid choices are `1` for differential, `2` for referenced single-ended, and `3` for non-referenced single-ended.

NIDAQDevice Error Messages

Error loading NI-DAQ driver
Lookout was not able to communicate to the NI-DAQ driver. Be sure that NI-DAQ has been installed properly, and the NI-DAQ Configuration Utility has been used to configure the your hardware devices.

NIDAQ: Analog Input: Invalid Data Member(s)
NIDAQ: Analog Output: Invalid Data Member(s)
NIDAQ: Digital Input: Invalid Data Member(s)
NIDAQ: Digital Output: Invalid Data Member(s)
At least one channels you have specified is not valid for your current hardware configuration. Be sure your hardware is configured properly using the NI-DAQ Configuration Utility.

NIDAQ: Error code: *NNNNN*
NIDAQ: Analog Input: Error code: *NNNNN*
NIDAQ: Analog Output: Error code: *NNNNN*
NIDAQ: Digital Input: Error code: *NNNNN*
NIDAQ: Digital Output: Error code: *NNNNN*
NI-DAQ has detected an error condition. Please refer to your *NI-DAQ Function Reference Manual* to determine the meaning of the error code.

Table 18-77. National Instruments Data Acquisition Devices Supported by Lookout

Device	AI	AO	DIO
AT-AO-6	—	yes	yes
AT-AO-10	—	yes	yes
AT-MIO-16E-1	yes	yes	yes
AT-MIO-16E-2	yes	yes	yes
AT-MIO-16E-3	yes	yes	yes
AT-MIO-16DE-10	yes	yes	yes
AT-MIO-16XE-50	yes	yes	yes

Table 18-77. National Instruments Data Acquisition Devices Supported by Lookout (Continued)

Device	AI	AO	DIO
AT-MIO-16X	yes	yes	yes
AT-MIO-16F-5	yes	yes	yes
AT-MIO-64E-3	yes	yes	yes
CS-2017	CJC	—	—
DAQCard-1200	yes	yes	yes
DAQCard-700	yes	—	yes
DAQCard-5XX	yes	—	yes
DAQPad-MIO-16XE-50	yes	yes	yes
DAQPad-1200	yes	yes	yes
Lab-PC+	yes	yes	yes
PC-LP-16	yes	—	yes
SC-2070	CJC	—	—
SCB-68	CJC	—	—
SCB-100	CJC	—	—

NISCXI

Lookout uses the NISCXI object class to communicate with National Instruments data acquisition devices connected in multiplex mode to SCXI hardware. Refer to the NIDAQDevice object class for more information concerning parallel SCXI configurations, and data acquisition configurations without SCXI.

To use this object, you should have NI-DAQ 5.0 or better software installed. While the NIDAQDevice object works with some earlier versions of NI-DAQ software, it does not perform well with these earlier versions, and has not been extensively tested.

Consult your NI-DAQ and NI-SCXI hardware, and NI-DAQ software manuals for information on installing and configuring National Instruments data acquisition software and hardware.

When you create an NISCXI object for the first time, or when you launch Lookout and run an application using the NISCXI object, Lookout automatically loads the NI-DAQ software. This can take a few moments, during which the screen is frozen. Once the NI-DAQ software has loaded, Lookout returns to its former speed.

☞ **Note** ***This object class is available on 32-bit versions of Lookout 3.7 and later. It is not backward compatible with earlier versions of Lookout, and does not run on the 16-bit version of Lookout.***

Figure 18-77. NISCXI Definition Parameters Dialog Box

Device is the NI-DAQ device number of the controlling device. The DAQ configuration utility (`WDAQConf.exe`) assigns this number to an installed device. Valid device numbers range is from 1 to 16.

On-board channel specifies the analog input channel of the controlling device used to address this SCXI chassis. When only one chassis is being controlled by the controlling board, this value should be 0. When this chassis in a multi-chassis configuration, this value should reflect the order of this chassis. For instance, the first chassis should be 0, the next chassis should be 1, and so on. Valid range is 0 to 7. If there are no analog input channels in this chassis, use the value 0.

SCXI chassis id is the chassis Id number assigned to this chassis by the DAQ configuration utility.

Scanning interval is the time period between analog and digital input polls. Valid range is 20 msec to 1 day (expressed in msec).

Communication alarm priority determines the priority level of the alarms generated by the NISCXI object.

NISCXI Data Members

Table 18-78. NISCXI Device Data Members

Data Member	Type	Read	Write	Description
CommFail	logical	yes	no	Object-generated signal that is on if, for whatever reason, Lookout cannot communicate with or control the device without error.
MD1.AI0–MD12.AI63	numeric	yes	no	Analog input channel in volts.
MD1.AO0–MD12.AO63	numeric	no	yes	Analog output channel in volts.
MD1.cjtemp–MD12.cjtemp	numeric	yes	no	Built-in cold-junction sensor on analog input terminal blocks.
MD1.DI0–MD12.DI31	logical	yes	no	Digital input line.
MD1.DO0–MD12.DO31	logical	no	yes	Digital output line.
MD1.Etc0–MD12Etc31	numeric	yes	no	Analog input channel with built-in polynomial conversion for an E type thermocouple.

Table 18-78. NISCXI Device Data Members (Continued)

Data Member	Type	Read	Write	Description
MD1.Ktc0 – MD12Ktc31	numeric	yes	no	Analog input channel with built-in polynomial conversion for a K type thermocouple.
MD1.Jtc0 – MD12Jtc31	numeric	yes	no	Analog input channel with built-in polynomial conversion for a J type thermocouple.
MD1.Rtc0 – MD12Rtc31	numeric	yes	no	Analog input channel with built-in polynomial conversion for an R type thermocouple.
MD1.Stc0 – MD12Stc31	numeric	yes	no	Analog input channel with built-in polynomial conversion for an S type thermocouple.
MD1.Ttc0 – MD12Ttc31	numeric	yes	no	Analog input channel with built-in polynomial conversion for a T type thermocouple.
Update	logical	yes	no	Object-generated signal that pulses low each time the object polls the device.

Configuring NIDAQ.INI for NISCXI

You may configure individual channel attributes by editing the `NIDAQ.INI` file in your Lookout directory.

Channel Attributes

To specify a channel, use the following format: `[DEV3.SC1.MD2.AI5]`

This specifies analog input channel 5 on module 2 of SCXI chassis 1 controlled by device 3.

UPPERLIMIT specifies the upper input limit in volts (for example, `UPPERLIMIT=2.5`). Usually used in conjunction with **LOWERLIMT**.

LOWERLIMIT specifies the lower input limit in volts (for example, `LOWERLIMIT=-3.7`). Usually used in conjunction with **UPPERLIMIT**.

Cold-Junction Sensor Attributes

To specify a cold-junction sensor, use the following format:
[DEV3.SC1.MD2.CJC]

This specifies the cold-junction sensor on module 2 of SCXI chassis 1 controlled by device 3.

SENSORTYPE specifies the type of temperature sensor on the terminal block. Valid choices are `IC` and `THERMISTOR`. Check your SCXI terminal block documentation to determine the correct sensor type.

RESAMPLE specifies the number of minutes between resampling the cold-junction sensor. Valid range is `1` to `10080`, and `0`. Use `0` to only check the cold-junction sensor once at the start, and do not resample the sensor later.

NISCXI Error Messages

Error loading NI-DAQ driver
Lookout was not able to communicate to the NI-DAQ driver. Be sure that NI-DAQ has been installed properly, and that you used the NI-DAQ Configuration Utility to configure the your hardware devices.

NIDAQ: Analog Input: Invalid Data Member(s)
NIDAQ: Analog Output: Invalid Data Member(s)
NIDAQ: Digital Input: Invalid Data Member(s)
NIDAQ: Digital Output: Invalid Data Member(s)
At least one channels you have specified is not valid for your current hardware configuration. Be sure your hardware is configured properly using the NI-DAQ Configuration Utility.

NIDAQ: Error code: *NNNNN*
NIDAQ: Analog Input: Error code: *NNNNN*
NIDAQ: Analog Output: Error code: *NNNNN*
NIDAQ: Digital Input: Error code: *NNNNN*
NIDAQ: Digital Output: Error code: *NNNNN*
NI-DAQ has detected an error condition. Please refer to your *NI-DAQ Function Reference Manual* to determine the meaning of the error code.

SCXI Devices

You can use the following National Instruments data acquisition devices with Lookout.

Device	AI	AO	DIO
SCXI-1000	—	—	—
SCXI-1001	—	—	—
SCXI-1100	yes	—	—
SCXI-1102	yes	—	—
SCXI-1120	yes	—	—
SCXI-1121	yes	—	—
SCXI-1124	—	yes	—
SCXI-1160	—	—	yes
SCXI-1161	—	—	yes
SCXI-1162	—	—	yes
SCXI-1163	—	—	yes
SCXI-1163R	—	—	yes
SCXI-1200	yes	yes	yes
SCXI-1300	CJC	—	—
SCXI-1303	CJC	—	—
SCXI-1320	CJC	—	—
SCXI-1328	CJC	—	—
SCXI-2000	yes	yes	yes
SCXI-2400	yes	yes	yes

Omron

Omron is a protocol driver class Lookout uses to communicate with Omron devices using the Host Link serial communication protocol.

An Omron object contains a great deal of data. It supports reading and writing of all predefined data points. When you create an Omron object, you have immediate access to all the data members for that object (see *data member* list below).

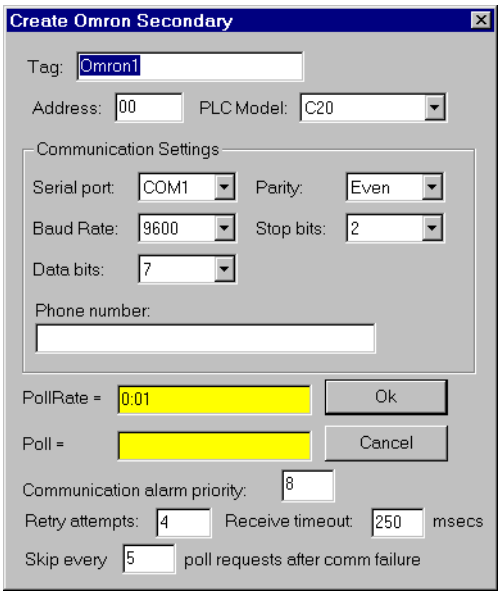


Figure 18-78. Omron Definition Parameters Dialog Box

Serial port specifies which COM port the object uses for communicating to the external device. This does not specify the communication type. Communication type is determined by the **Options»Serial Ports…** command.

Data rate indicates the baud rate that Lookout uses to communicate with the hardware device. This **Data rate** setting should match the selection made on the physical device.

Data bits indicates the number of data bits that Lookout uses to communicate with the hardware device. This **Data bits** setting should match the selection made on the physical device.

Stop bits indicates the number of stop bits that Lookout uses to communicate with the hardware device. This **Stop bits** setting should match the selection made on the physical device.

Parity indicates the parity that Lookout uses to communicate with the hardware device. This **Parity** setting should match the selection made on the physical device.

Phone number specifies the number to be dialed if the selected serial port is configured for dial-up. This **Phone number** only applies to the individual protocol object.

PollRate is a numeric expression that determines how often to poll the device. The object then polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *Introduction*, for more information on entering time constants.

Poll is a logical expression. When this expression changes from FALSE to TRUE, Lookout polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Communication alarm priority determines the priority level of alarms generated by the Omron object. Such alarms are typically related to communications with the physical device.

Retry attempts specifies the consecutive number of times Lookout attempts to establish communications with a device when it is not getting a valid response. After it tries the number of **Retry attempts** specified, the Omron object generates an alarm and releases the communication port back to the communications subsystem. The subsystem then moves on to the next device in the polling queue (if any). See Chapter 6, *Serial Communications*, for more information.

Receive timeout is the time delay Lookout uses in waiting for a response from a device before retrying the request.

The **Skip every…** setting instructs Lookout not to poll a device it has lost communication with on every scheduled poll. Instead, Lookout skips the device in the polling cycle. Once communications have been reestablished, the device is polled on its regular cycle.

Omron Data Members

As with all Lookout drivers, you can access I/O points and other data through data members. The following is a table of data members currently supported by the Omron object class.

Table 18-79. Omron Data Members

Data Member	Type	Read	Write	Description
AR0–AR27	numeric	yes	yes	Auxiliary relay area, read as 16-bit word.
AR0.0–AR27.15	logical	yes	yes	Auxiliary relay area, read as 1-bit discrete.
DM0–DM9999	numeric	yes	yes	Data memory area, 16-bit word.
CommFail	logical	yes	no	Object-generated signal that is on if, for any reason, Lookout cannot communicate with the device(s).
HR0–HR99	numeric	yes	yes	Holding relay area, read as 16-bit word.
HR0.0–HR99.15	logical	yes	yes	Holding relay area, read as 1-bit discrete.
IR0–IR511	numeric	yes	yes	I/O area, read as 16-bit word.
IR0.0–IR511.15	logical	yes	yes	I/O area, read as 1-bit discrete.
LR0–LR63	numeric	yes	yes	Link relay area, read as 16-bit word of information.
LR0.0–LR63.15	logical	yes	yes	Link relay area, read as 1-bit discrete.
TC0–TC999	numeric	yes	no	Timer/Counter, read as 16-bit word.
Update	logical	yes	no	Object-generated signal that pulses low each time it polls the device.

Note ***The Omron requires a special cable configuration in order to work properly. See your Omron hardware documentation for the correct configuration.***

Omron Status Messages

No response within timeout period
Lookout received no response from a device within the **Receive timeout** period. The Omron object is able to use the COM port, but when it polls the device, it does not respond—as if it is not even there.

Cannot set PLC to MONITOR mode
The Omron object is trying to set the PLC in MONITOR mode in order to communicate with the PLC correctly, but cannot perform the operation.

Incorrect address in response
The frame received had an incorrect source address. Check for two or more devices with the same address.

Incorrect command in response
The frame received had an incorrect command. Check for two or more devices with the same address.

Incorrect data type in response
The frame received had an incorrect data type marker.

Incorrect frame check sum (FCS)
The frame received had an incorrect check sum.

Omron errors reported in the response
These errors are reported by the Omron device, and are in turn reported to you in text form.

Omron Models Supported

C20, C200, C500, C1000, C2000, CQM, CPM1

OneShot

OneShot is an adjustable delay timer. When **On/off signal** transitions to on, the output signal goes TRUE and the **Timer delay** begins to count down. At the end of the delay countdown, the output signal goes FALSE.

Unlike the Interval timer, the OneShot timer output remains on for the **Time delay** period even if **On/off signal** goes FALSE. So a OneShot timer requires only a momentary signal to begin the **Timer delay** period. Pulsing the **On/off signal** during the time delay period does not extend the time delay period of a OneShot timer.

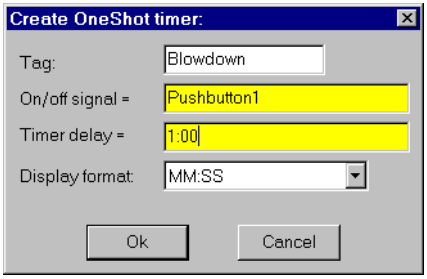


Figure 18-79. OneShot Definition Parameters Dialog Box

The **On/off signal** is a logical expression while Timer delay is a numeric expression. Normally, this is a simple time constant such as 0:20 (twenty seconds). See *Numeric Data Members* in See Chapter 5, *Developer Tour*, for information on time constants.

Timer delay can range from 0.0 seconds to several years. The effective resolution is 0.1 seconds over the entire range.

The object is represented on a control panel by showing the time delay remaining in the format defined by the **Display format** parameter. It is updated approximately once per second. If the delay period has expired, it shows **OFF**.

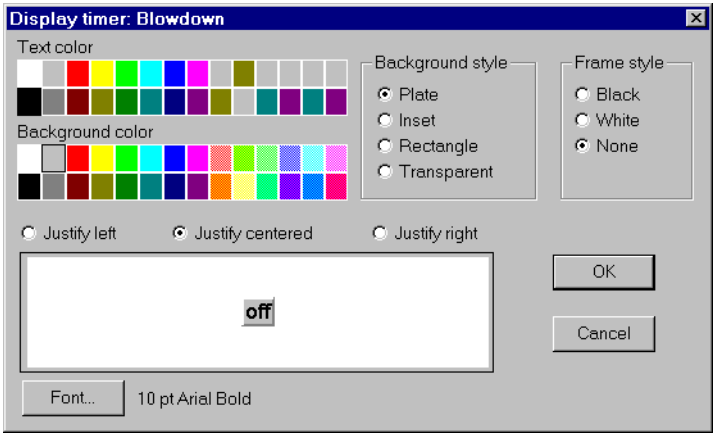


Figure 18-80. OneShot Display Parameters Dialog Box

OneShot Data Members

Table 18-80. OneShot Data Members

Data Member	Type	Read	Write	Description
(implicit)	logical	yes	no	Logical timer value

Comments ***You can use the OneShot timer to hold a valve open for a set period of time after a pushbutton is pressed, or to prevent pump starts from occurring too rapidly in succession.***

Related Objects ***DelayOff, DelayOn, Interval, Pulse, TimeOfxxx***

Optomux

Optomux is a protocol driver class Lookout uses to communicate with Opto 22 I/O devices and other equipment using the Optomux communication protocol.

This protocol driver uses Optomux 2-pass protocol with no parity, eight data bits, and zero stop bits. You can choose between CRC16 and BCC error correction modes, and select a data rate.

Several physical Optomux devices can be daisy-chained or multidropped on one ASCII network. In Lookout, you create a single Optomux object to represent all devices connected to the same COM port.

An Optomux object contains a great deal of data. It represents all devices and their associated channels. It supports reading and writing of all predefined data point types including analog input, discrete input, analog output, discrete output, counter, latch, latch clear, and temperature. When you create an Optomux object, you have immediate access to all the object data members (see data member list below).

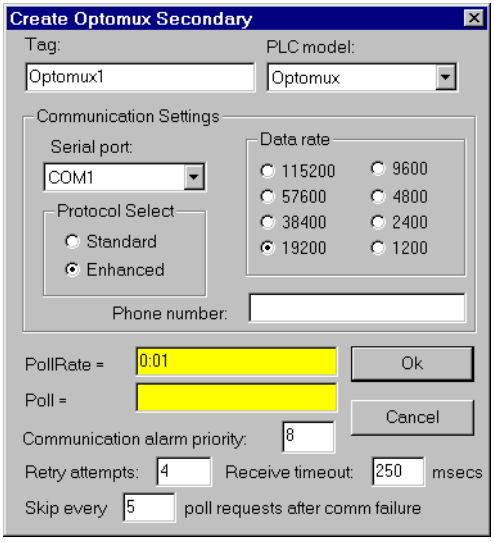


Figure 18-81. Optomux Definition Parameters Dialog Box

Serial port specifies which COM port the object uses for communicating to the external device. This does not specify the communication type. Communication type is determined by the **Options»Serial Ports…** command.

Protocol Select specifies the type of error correction mode used by the RTU. **Enhanced** mode uses CRC16 error checking. **Standard** mode uses BCC (horizontal parity). **Enhanced** mode is the default and is recommended.

Data rate indicates the baud rate that Lookout uses to communicate with the hardware device. This setting should match the selection made on the physical device.

Phone number specifies the number to be dialed if the selected serial port is configured for dial-up. This number only applies to the individual protocol object.

PollRate is a numeric expression that determines how often to poll the device. The object then polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *Introduction*, for information on entering time constants.

Poll is a logical expression. When this expression changes from FALSE to TRUE, Lookout polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Communication alarm priority determines the priority level of alarms generated by the Optomux object. Such alarms are typically related to communications with the physical device.

Retry attempts specifies the consecutive number of times Lookout attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the Optomux object generates an alarm and releases the communication port back to the communications subsystem, which then moves on to the next device in the polling queue (if any). Refer to the communications chapter for more information.

Receive timeout is the time delay Lookout uses in waiting for a response from a device before retrying the request.

The **Skip every…** setting instructs Lookout to not poll a device it has lost communication with on every scheduled poll. Instead, Lookout skips the device in the polling cycle accordingly. Once communications have been reestablished, the device is polled on its regular cycle.

Optomux Watchdog Capability

The Optomux protocol has a built in watchdog capability. The device monitors communications at all times. If the device receives no communications within the amount of time specified by `WDTime`, the device goes into *watchdog state*. This state makes sure all your devices are in a fail-safe condition if there is any kind of COM failure.

The DWD and AWD data members write to the device the values that device will use if it enters the watchdog state. They are write only.

Notice that the DWD data members are not logical values, as you might expect them to be. They also do not have channel numbers. This is because you set the watchdog data of an entire discrete module with one command and one value. Because there may be 16 channels possible, the single value sent may be as large as 16-bit.

For example, if you want to set the watchdog data for the discrete output module (DWD) at address 3, perform the following steps:

1. Connect a numeric value, such as 10 seconds, to `WDTime`. There has to be something connected to data member or you cannot set watchdog data. WDTime reads the numeric input as milliseconds; to set the watchdog time to 10 seconds, the numeric input must be 10000.
2. If you want Channels 1,2, and 16 to be ON when the device goes into watchdog state, and all other channels to be OFF, send the hex value `0x8003` to `DWD3`. This expands to 1000 0000 0000 0011 in binary (Optomux arranges channels from left to right in the binary sequence). You can use the **Change»Numeric Format**... option to create a hexadecimal numeric in Lookout.

AWD data members work the same way, except that you have a separate data member for each channel, and you give it a numeric value. To set channel 3 of the analog output module at address 10, you would connect the numeric value to `AWD10.3`.

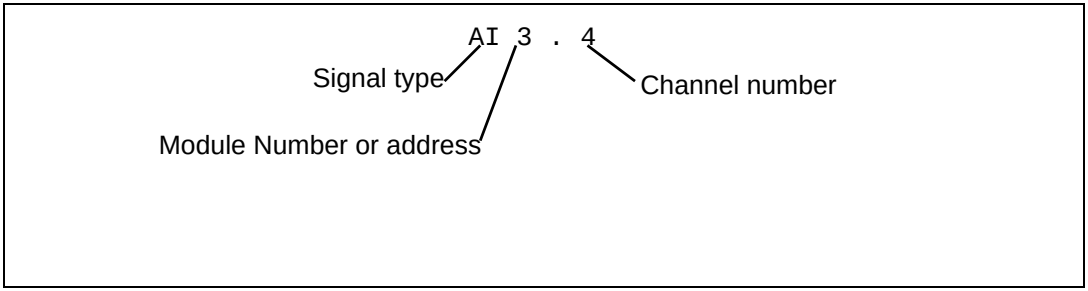
Optomux Data Members

As with all Lookout drivers, you can access I/O points and other data through data members. The following is a table of data members currently supported by the Optomux object class.

Table 18-81. Optomux Data Members

Data Member	Type	Read	Write	Description
AI0.1 – AI255.16	numeric	yes	no	Analog input. 12 bit word containing an unsigned integer ranging from 0 to 4095
AO0.1 – AO255.16	numeric	yes	yes	Analog output. 12 bit word containing an unsigned integer ranging from 0 to 4095
AWD0.1 – AWD255.16	numeric	no	yes	Analog watchdog data
CNT0.1 – CNT255.16	numeric	yes	no	Counter. Returns accumulated count for specified inputs
CommFail	logical	yes	no	Object-generated signal that is on if, for whatever reason, Lookout cannot communicate with the device(s)
DI0.1 – DI255.16	logical	yes	no	Discrete input
DO0.1 – DO255.16	logical	yes	yes	Discrete output
DWD0 – DWD255	numeric	no	yes	Discrete watchdog data
LAT0.1 – LAT255.16	logical	yes	no	Returns ON status for inputs which are latched
LTC0.1 – LTC255.16	logical	yes	no	Returns ON status for latched input and clears the latch
Update	logical	yes	no	Object-generated signal that pulses low each time it polls the device
WDtime	numeric	no	yes	Watchdog time (in ms)

Comments ***To specify the address of an analog input located on channel 4 of module 3, you would enter*** `AI3.4`***, where***



☞ **Note** ***The Optomux protocol may perform limited value-type verifications depending upon the hardware you use. In other words, trying to read analog input data from a discrete module may be allowed but the data returned may be arbitrary (or zero).***

Optomux Status Messages

No response within timeout period
Lookout received no response from a device within the **Receive timeout** period. The Optomux object is able to use the COM port, but when it polls the device, it does not respond—as if it is not even there. If you have daisy-chained several devices, you have introduced an inherent delay. You may have to significantly increase **Receive timeout** (and **Poll Rate**) to ensure Lookout is allowing enough time to receive the expected response. This increase has nothing to do with the processing capabilities of Lookout. Rather it is based solely on **Data rate** and the number of devices on the chain. Also, verify your baud rate settings, cable connections, power, configuration settings, COM port settings, and polling addresses.

Invalid data request from module
Either you tried to read from a module that is missing, or you attempted a data type mismatch. For example, you may have tried to read a discrete value from an analog module. This data type mismatch is seen on a data analyzer as a frame containing question marks. This may be an addressing problem. Verify that the type of I/O at the desired channel matches the signal type of the address identified in the data member you are writing to.

Message garbled – Bad CRC or BCC
The Optomux object is receiving a poll response from the device(s), but it could not decipher the response. Verify that the error correction scheme you selected in **Protocol Select** is the same as that of all of the physical devices. Also verify that all devices connected to the COM port have unique addresses. It may also be that the last part of the message is getting clipped off before it is completed. Consider increasing the number of **Retry attempts**. You may have to increase the **Receive gap** Serial Port setting to

ensure Lookout is receiving the entire message. If your Serial Port is configured for radio this could be caused by an audible squelch tail occurring at the end of a radio transmission. You may need to adjust the **RTS delay off** or the **CTS timeout** settings.

Unexpected data response length
The Optomux object is receiving a poll response from the device(s), but the response is too long. Verify that all devices connected to the COM port have unique addresses. Devices with identical addresses may be trying to respond at the same time.

Response too short
The Optomux object is receiving data that does not meet the minimum frame length requirements. This could happen if you configured a device for 4-pass instead of 2-pass protocol. Lookout uses 2-pass protocol. Verify device protocol settings.

Pager

Pager is an object class Lookout uses to contact a numeric or alphanumeric pager through a modem, sending a message to the pager.

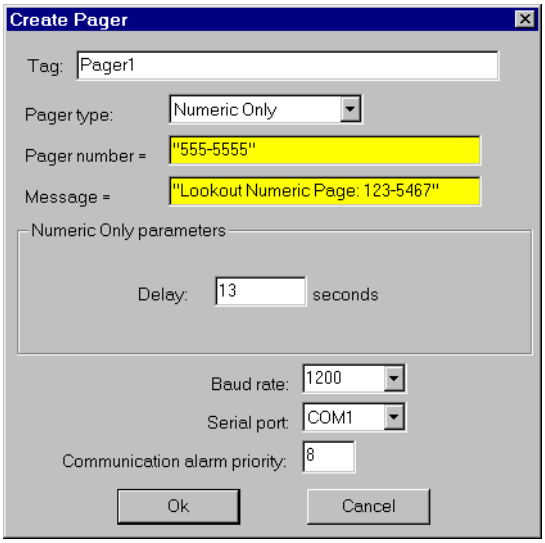


Figure 18-82. Numeric Only Pager Definition Parameters Dialog Box

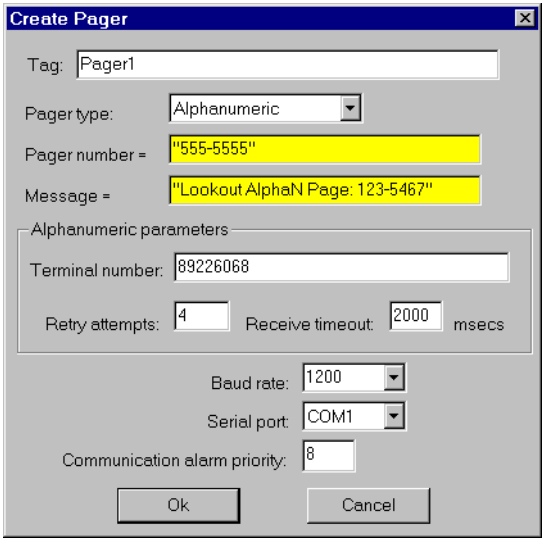


Figure 18-83. Alphanumeric Pager Definition Parameters Dialog Box

Pager type determines whether the Pager object operates in numeric only or alphanumeric mode. A detailed description of the operation of these two modes follows.

Pager number is the phone number of the pager you want to contact. When the Pager object is in **Alphanumeric** mode, this number corresponds to the pager ID number.

Message is the message you want to send to the pager. Notice that in **Numeric Only** mode only numeric characters are sent.

Delay is how long the Pager object waits after dialing the pager number before it dials the message number. This parameter is valid in **Numeric Only** mode only.

Terminal number is the phone number of the remote paging terminal you want to contact. This parameter is valid in **Alphanumeric** mode only.

Retry attempts specifies the consecutive number of times Lookout attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the Pager object generates an alarm and releases the COM port. Refer to Chapter 6, *Serial Communications,* for more information. This parameter is valid in **Alphanumeric** mode only.

Receive timeout is the time delay Lookout uses in waiting for a response from a device before retrying the request. This parameter is valid in **Alphanumeric** mode only.

Baud rate indicates the baud rate Lookout uses to communicate with the modem and paging terminal.

Serial port specifies which COM port Lookout uses to communicate with your modem. You must have this COM port configured as dial-up under **Options»Serial Ports**.

Communication alarm priority determines the priority level of alarms generated by the Pager object. You can relate such alarms to communications with the modem or with the remote paging terminal.

Pager Data Members

Table 18-82. Pager Data Members

Data Member	Type	Read	Write	Description
Send	logical	no	yes	Sends the message on transition from FALSE to TRUE
Message	text	no	yes	Pager message
Phone	text	no	yes	Individual pager phone number or Page ID number

Pager Object Modes

Numeric Only

In **Numeric Only** mode, the Pager object establishes a connection with your local modem. Once this connection has been established and the pager number dialed, the Pager object waits for the time specified by **Delay**, then dials the number that is the message. Because the `Message` data member is a text value, the Pager object in **Numeric Only** mode omits any non-numeric characters from the message when it is sent.

Alphanumeric Mode

In **Alphanumeric** mode, the Pager object actually establishes a connection with a remote paging terminal, then transmits an alphanumeric message using Telocator Alphanumeric Protocol (TAP) version 1.8. TAP is an industry standard protocol for paging terminals that accept alphanumeric pages. Alphanumeric messages are limited to 250 characters. The text value in the `Message` data member will be truncated to this length if it is longer.

Pager Serial Port Settings

Notice that there are two different retry settings that affect the operation of the Pager object in **Alphanumeric** mode. The retry settings in the Pager object dialog box govern serial communications with the remote paging terminal. This means that after the two modems have connected and finished handshaking, and the serial transaction is underway, each individual frame is timed by the **Receive timeout** setting, and retried the number of times specified by **Retry attempts**.

These retry settings will not dial the phone number again if the remote paging terminal for some reason does not answer or is busy, which happens occasionally. This setting and other important modem settings (including the AT initialization string that the Pager object must use on your modem) can be found in **Option»Serial Ports**, and should be chosen carefully. These settings are important in both Pager object modes.

☞ **Note** ***You may have to increase your Receive Gap setting from its default of 5 to something closer to 20 or 25. You must also have your COM port configured as dial-up.***

Pager Queueing

The Pager object queues up to 10 messages in either mode. If the object is in the process of sending out a page and the `Send` data member goes high again, the current value of the `Message` data member will be queued and sent out when the Pager object has time. Messages that are already in the queue will not be duplicated.

Pager Status Messages

Paging terminal refused logon
Alphanumeric only error code

Paging terminal forced disconnect
Alphanumeric only error code

Paging terminal NAKed block transmission
Alphanumeric only error code

Paging terminal abandoned block transmission
Alphanumeric only error code

No response within timeout period
This means that the modem is not responding to Lookout requests.

Queue full
The paging queue currently has 10 pages in it, and will not accept any more until at least one of those pages is successfully sent.

Garbled response
A response from the modem was corrupted or in an unrecognizable form.

Panel

Panel is a unique object class that accepts display members of other objects. Panels (Control panels) are a window into your process you can use to monitor and control your system by flipping switches, depressing pushbuttons, and turning knobs.

There is no limit as to the number of control panels you can create in Lookout or the number of objects or graphics that you can display on a given panel. Control panels can be any size, and you can display them within the Lookout workspace in various states (maximized, normal, minimized).

Create control panels with the **Object»Create…** command or with the **Insert»Control panel…** command. Both commands deliver the same result.

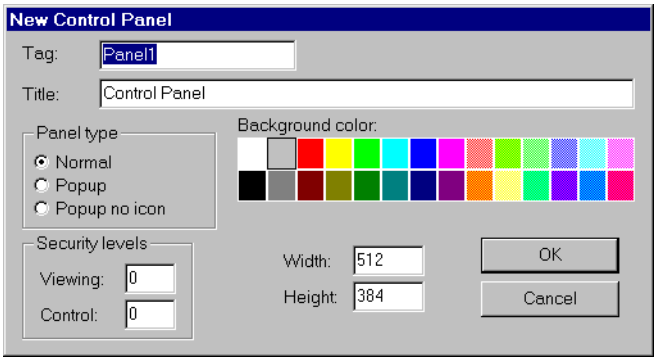


Figure 18-84. Panel Definition and Display Parameters Dialog Box

There are three distinct panel types: **Normal**, **Popup**, and **Popup no icon**.

A **Normal** control panel can be maximized, normal size, or minimized within the Lookout workspace. When you activate a **Normal** panel it appears at the size defined by its **Height** and **Width**. When you maximize a **Normal** panel, it fills the Lookout workspace. When you minimize a **Normal** panel it appears as an icon. The **Normal** option is typically selected for full-sized control panels.

Popup control panels can either be in a popup state or minimized (they cannot be maximized). When you activate a **Popup** panel it appears at the size defined by its **Height** and **Width**. When a Popup control panel is activated, it remains on top of all other panels until it is minimized. When

you minimize a **Popup** panel it appears as a bar at the bottom of the Lookout screen.

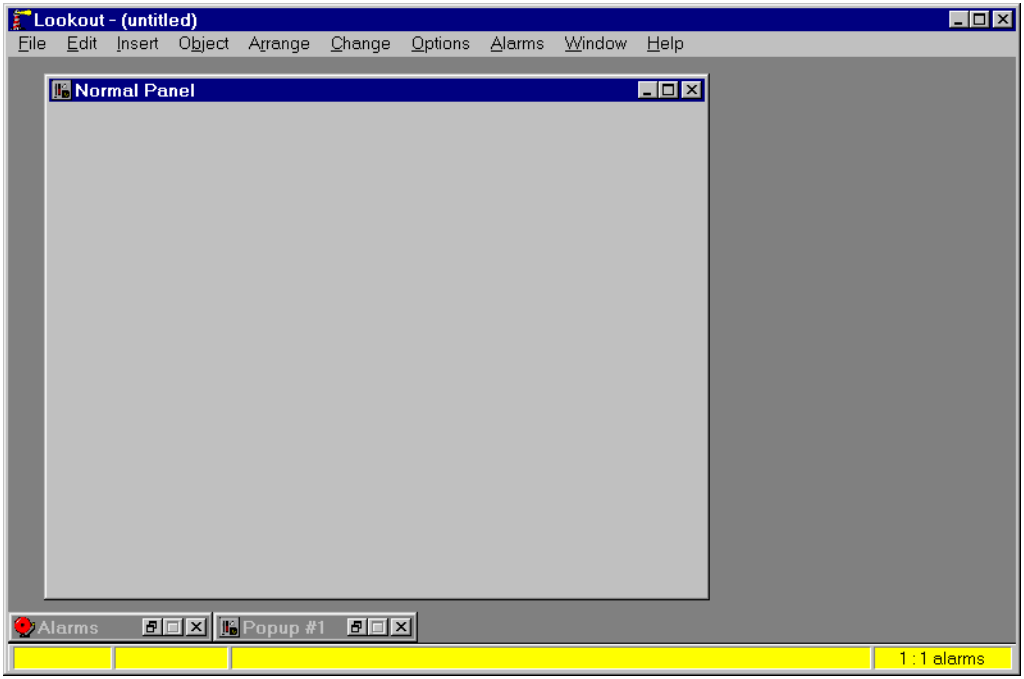
Popup no icon control panels are identical to **Popup** panels except they are not represented by icons when minimized. When you minimize a **Popup no icon** panel it disappears from the Lookout workspace.

☞ **Note** ***Because*** *Popup no icon* ***control panels are not represented by icons, they use less memory. (Microsoft Windows allocates a fixed amount of memory to each icon.) If you are experiencing memory problems when running a Lookout application that has many control panels, consider converting your*** *Popup* ***panels to*** *Popup no icon****.***

Normal panels and Popup panels can be chosen by selecting their icon, using the **Window** menu command. **Popup no icon** control panels cannot; they can only be accessed by triggering their `activate` data member.

The **Security levels** settings are globally applied to a given control panel. The **Control** security level works in conjunction with all individual object security levels on that panel. The higher security level of the two is used to determine if an operator has control over the object. For example, consider a single Switch object with a security level of 4 that is displayed on two panels. The first panel has a control level of 6 and the second panel has a control level of 2. Only level 6 and higher operators are able to flip the switch on the first panel; however, level 4 and higher operators have control over the same switch on the second panel.

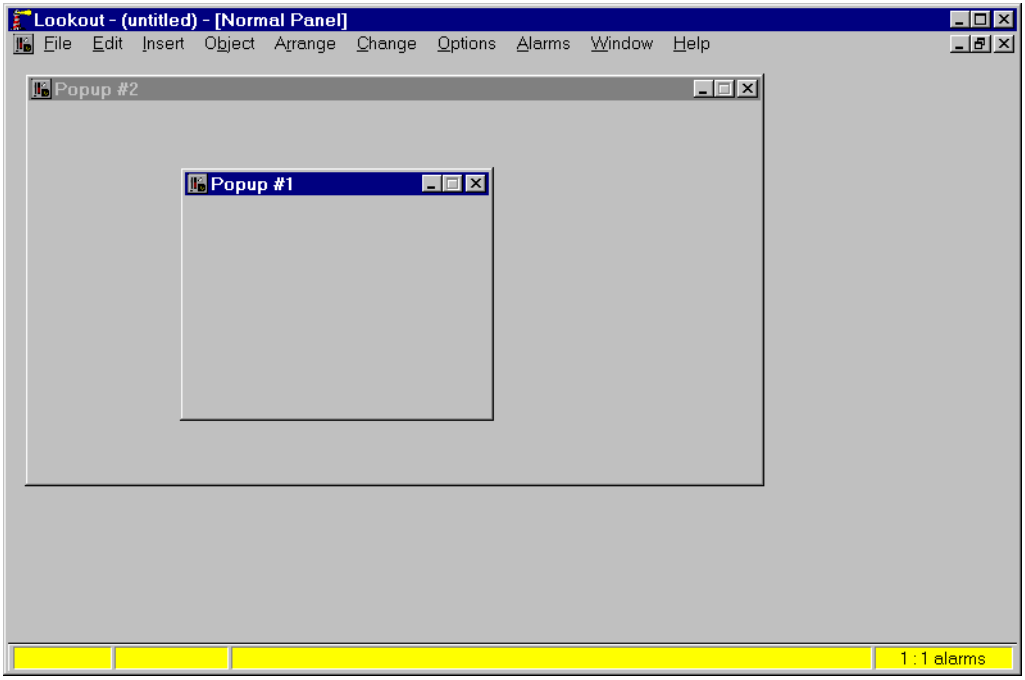
Viewing security can hide entire control panels from low level operators. This parameter affects the entire control panel. As an example take a control panel with a viewing security level of 6. If a level 5 (or lower) operator logs on, he is unable to see the control panel. In fact, he does not even know it exists because it is not listed in the **Window** menu and it is not shown as an icon. If a level 6 (or higher) operator logs on, the control panel instantly becomes available for display. This feature is useful for hiding panels that are rarely used or that contain sensitive information.



This example shows a **Normal** control panel in a normal state and a **Popup** control panel in a minimized state. The **Normal** panel could easily be minimized or maximized by hitting the appropriate arrow button.

The following example is a typical scenario involving full screen control panels with multiple popup panels displayed at once.

This configuration maximizes the amount of information that you can display at once; and it allows you to have any number of different combinations of control panels displayed on your monitor.


Lookout - (untitled) - [Normal Panel]
File Edit Insert Object Arrange Change Options Alarms Window Help
Popup #2
Popup #1
1 : 1 alarms

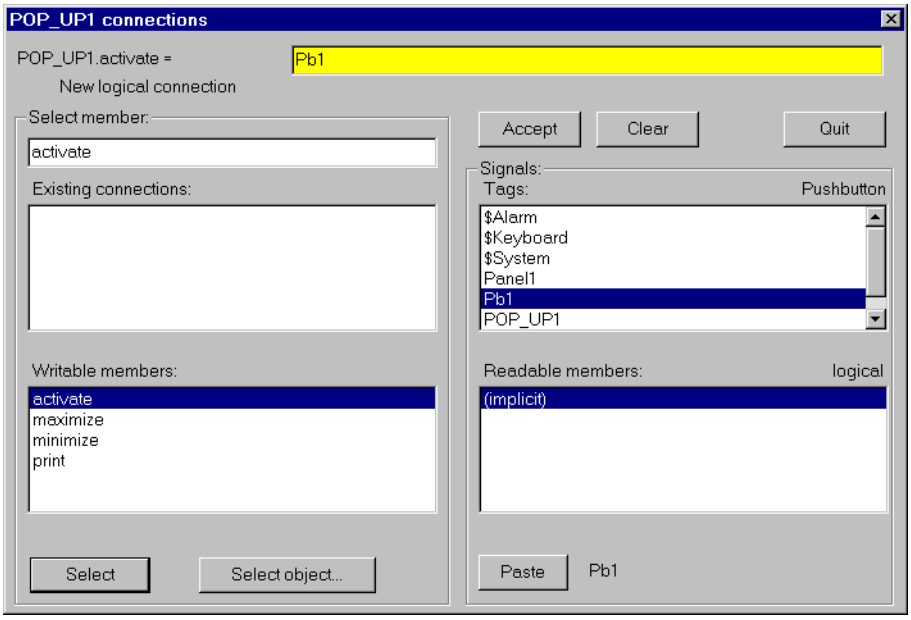
This example displays a **Normal** control panel in a maximized state and two **Popup** control panels in a "popped up" state.

Manipulating Panels

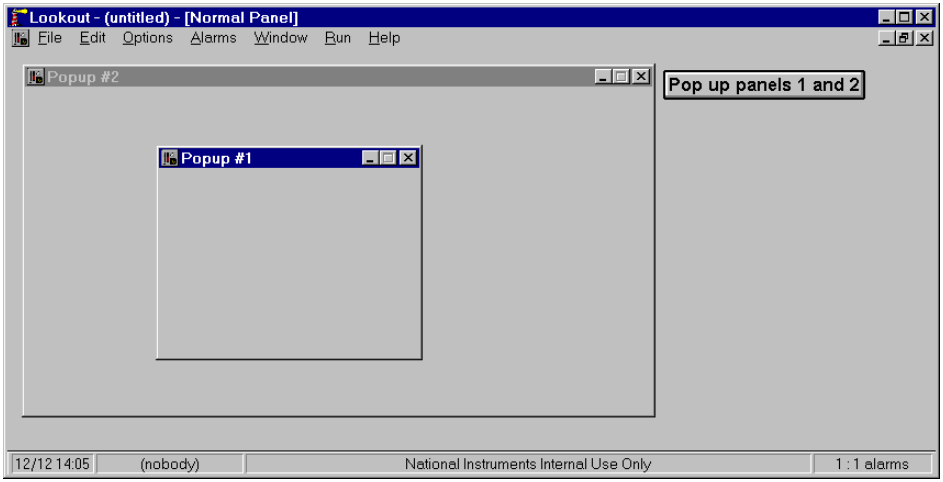
Lookout control panels utilize the Microsoft standard Multiple Document Interface (MDI) techniques. You manipulate Lookout windows the same way you do other windows in the Microsoft Windows environment.

Panel Switching

It is common to have several or even dozens of control panels. Creating a methodology for moving between your panels can be as simple or as elaborate as you want. One effective method utilizes pushbuttons that invoke other control panels. You connect the pushbutton output signal to the activate or maximize data member of the control panel(s) you want to affect. When the button signal goes high the respective panel(s) appear. The following example shows a single pushbutton and X with its output signal connected to two **Popup** control panels. The pushbutton is inserted on a third maximized panel. See **Object»Edit Connections…** command in Chapter 17, *Edit Mode Menu Commands*, for additional information on connecting data members to signals/expressions.

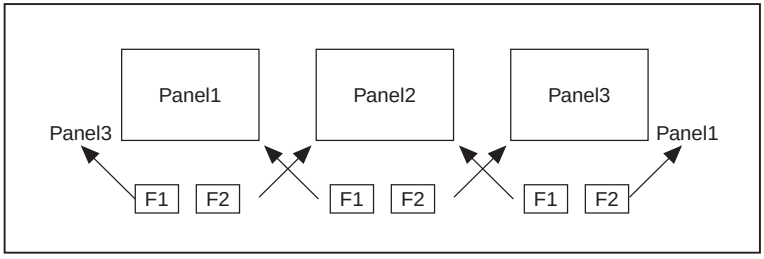


When you toggle out of Edit Mode and depress the button, both **Popup** control panels instantly appear, as shown below. Of course you could have connected the pushbutton to another **Normal** panel instead, and it would have appeared as the new maximized panel.



As you can imagine, there is no limit to the number of connections between various signals and control panels. In fact, you can create complex expressions/alarms that automatically call up specific control panels.

Another way to move between panels is through the use of function keys. Like the $Keyboard object, each panel has its own set of data members representing the function keys F1 – F24. The following example shows the F2 data member of Panel1 connected to the activate data member of Panel2. The F2 data member of Panel2 is also connected to the activate data member of Panel3. This way an operator can depress F2 to page forward through several panels.



In similar fashion, the F1 data member of Panel3 is connected to the activate data member of Panel2 and the F1 data member of Panel2 is connected to the activate data member of Panel1. So now, `Panel2.activate = Panel1.F2 OR Panel3.F1`. An operator can depress F2 to page forward through the control panels and F1 to page back through the control panels.

Special Considerations for “Home Panel”

To ensure users do not get lost when switching between panels, you might define one panel as your master control panel, or home panel or computer main menu. You could connect the activate data member of your home panel to $Keyboard.Shift.F1, or perhaps to a pushbutton object. If connected to the function key, any time the user presses <SHIFT-F1> (no matter what panel he or she is looking at), the home panel is called, returning the operator to a familiar control panel.

You might also want the home panel to maximize upon startup. If you have already created a pushbutton to call the home panel, you can connect it to the `maximize` data member.

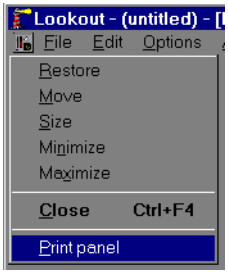
The exclamation point (!) instructs Lookout to use the opposite of the pushbutton value. At startup, the pushbutton is not depressed so its value is FALSE. But because you are using the opposite of the pushbutton value, Panel1.maximize is TRUE at startup. Any time a user depresses `CallHomePb` after this connection is made, nothing happens until the pushbutton is released—at which time the panel is called.

Panel Print

You can easily print a control panel using the **Print Panel** command. This function works equally well for both Normal and Popup panel types.

Print the contents of a panel by clicking on the panel control menu and then on **Print Panel**, or by connecting a logical expression to the `Print` data member of the desired control panel. A panel does not have to be visible to be printed.

☞ **Note** ***Certain metafiles look different on the printed page than they do on the display screen. This means that parts of layered objects sometimes appear opaque on the screen, but translucent when drawn on paper.***



☞ **Note** ***When you print a*** *Normal* ***panel, it is printed at its defined*** *Height* ***and*** *Width* ***parameters. If you define a panel whose*** *Height* ***and*** *Width* ***are at the default 400 × 300 pixel setting, maximize the panel, and then add graphic elements to the full panel, those elements outside of the default 400 × 300 pixel range are not shown when the panel is printed. To print all the elements on a maximized panel, modify the*** *Width* ***and*** *Height* ***of the panel to match the full-screen dimension of the panel.***

You can modify an existing control panel by toggling into edit mode and right-mouse clicking on its title bar. You can also modify a panel with the **Object»Modify…** menu command just like any other object.

Screen Resolution and Lookout Graphics

Lookout graphics and control panels appear different, depending on the PC you are using and the resolution of your screen driver. When you position a graphic (or any other display element) onto a control panel, Lookout identifies the position you selected by recording the specific pixel position of the graphic. (A pixel is the smallest possible dot on the screen.) Lookout actually counts the number of pixels that the graphic is from the upper left-hand corner of the screen. When you subsequently recall a panel, Lookout knows the exact location to place the graphic.

The reason to bring this up is because different computer screen drivers have different screen resolutions. VGA screens are 640 × 480 pixels. Super VGA screens typically range from 800 × 600 pixels to 1024 × 768 pixels. A panel created at 640 × 480 pixel resolution does not fill the screen of a 1024 × 768 super VGA monitor. A panel created at 1024 × 768 pixel resolution will overflow the screen of a 640 × 480 VGA monitor.

It is best to create your panels using the display driver resolution of the computer on which you intend to run Lookout. If you are creating panels for use on multiple computers, consider developing panels using the display driver resolution of the most common resolution monitor (if you have a dozen Super VGA computers and one VGA computer, develop your panels in Super VGA, not VGA). You will then have to modify the panels slightly to fit on the less common resolution computer(s).

You can usually change the resolution of your screen from VGA to Super VGA by changing System Settings in Windows Setup. Refer to your *Microsoft Windows* manual for more information.

Panel Data Members

Table 18-83. Panel Data Members

Data Members	Type	Read	Write	Description
active	logical	yes	no	Returns TRUE when the panel is the currently selected panel (i.e., it is active).
activate	logical	no	yes	Upon transition from FALSE to TRUE, calls control panel to focus or pop up.

Table 18-83. Panel Data Members (Continued)

Data Members	Type	Read	Write	Description
maximize	logical	no	yes	Upon transition from FALSE to TRUE, maximizes control panel, replacing existing maximized control panel.
minimize	logical	no	yes	Upon transition from FALSE to TRUE, minimizes control panel to icon state.
<F1> – <F24>	logical	yes	no	Each of these 24 data members represent a function key, <F1> – <F24>. Returns TRUE when the panel is active and its associated function key is pressed.
<SHIFT-F1> through <SHIFT-F24>	logical	yes	no	Each of these 24 data members represent a function key, <F1> – <F24>—when pressed in conjunction with the <SHIFT> key. Returns TRUE when the panel is active and its associated function key is pressed in conjunction with the <SHIFT> key.
<CTRL-F1> – <CTRL-F24>	logical	yes	no	Each of these 24 data members represent a function key, <F1> – <F24>—when pressed in conjunction with the <CTRL> key. Returns TRUE when the panel is active and its associated function key is pressed in conjunction with the <CTRL> key.
print	logical	no	yes	Upon transition from FALSE to TRUE, sends the control panel to the Windows Print Manager.

Pareto

The Pareto object class is one of the Lookout Statistical Process Control (SPC) tools and can play an important role in your Total Quality Management (TQM) program. This object class displays a frequency distribution of occurrences sorted by category so that you can identify the most frequently occurring anomalies or defects in your process.

The Pareto object class displays weighted and/or unweighted Pareto charts. The charts can be interactive, automatic, or a combination of both. An optional percentage line, showing cumulative percentage of factors, may be displayed. The number of factors to display, which may be less than the total number of factors defined, is user selectable. The chart background color, as well as the label, weight, and color of each factor, can be defined by the user.

The Pareto chart object definition dialog box is shown below.

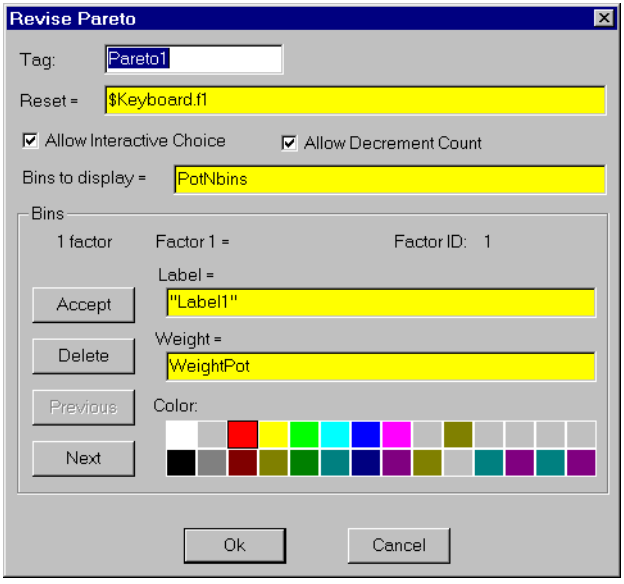


Figure 18-85. Pareto Definition Parameters Dialog Box

Reset is a logical expression that, on transition from FALSE to TRUE, resets all of the factor counts to zero.

Use **Allow Interactive Choice** to indicate whether the operator can increment (and possibly decrement) factor counts by clicking the mouse cursor on the Pareto chart object.

Allow Decrement Count indicates whether the operator can decrement the factor counts if you enable interactive choice.

Bins to display is the number of factor bins to display in the Pareto chart. This can be a numeric expression or a constant.

Bins parameters enable adding, modifying, or deleting factors to and from the Pareto chart. **Label** is the descriptive title for the factor. It can be a text constant or expression. **Weight** is the bias you can apply for a factor if you are going to display a weighted Pareto chart. It can be a numeric constant or expression. **Color** is the color of the bar for the factor. **Factor ID** is a number generated by the object to uniquely identify the current factor count when it is being saved to or restored from the Lookout state file. It is a visual indication to the user that a factor count is associated with a particular Factor ID, and not with the Label.

The Pareto display dialog box is shown below.

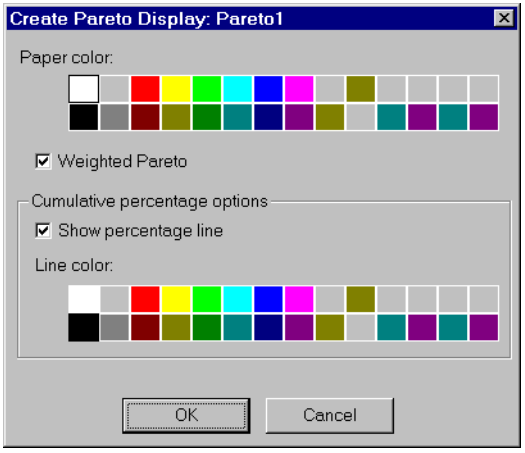


Figure 18-86. Pareto Display Parameters Dialog Box

Show percentage line determines whether a cumulative percentage line will be displayed in the color selected by **Line color**.

Weighted Pareto determines whether the Pareto is displayed as a weighted chart or an unweighted chart.

Weighted or Unweighted Charts

Using the same Pareto object, you can display weighted or unweighted Pareto charts. An unweighted Pareto chart shows which factors occur most frequently. If the factors are of approximately equal importance, an unweighted chart is a good indication of trouble spots.

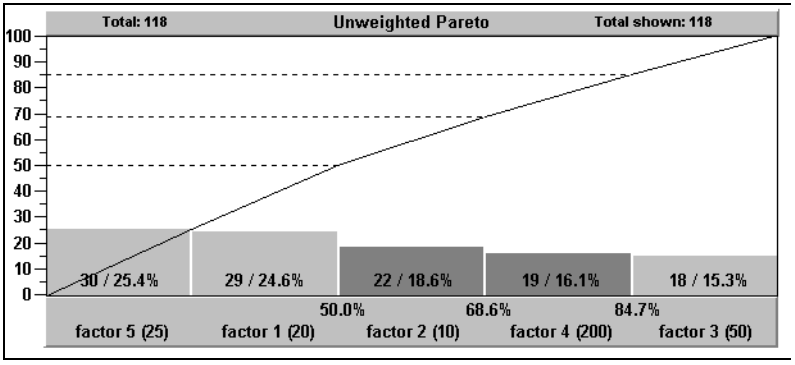


Figure 18-87. Unweighted Pareto Chart

If some factors are much more serious than others, using a weighted Pareto chart to ascertain the cost or exposure of a factor is a much more effective tool for identifying problem areas. For example, in the unweighted Pareto chart above, factor 1 and factor 5 together account for 50 percent of the defect occurrences. However, as you can see on the weighted Pareto chart below, factor 4 alone accounts for 60.8 percent of the total expense of all defects.

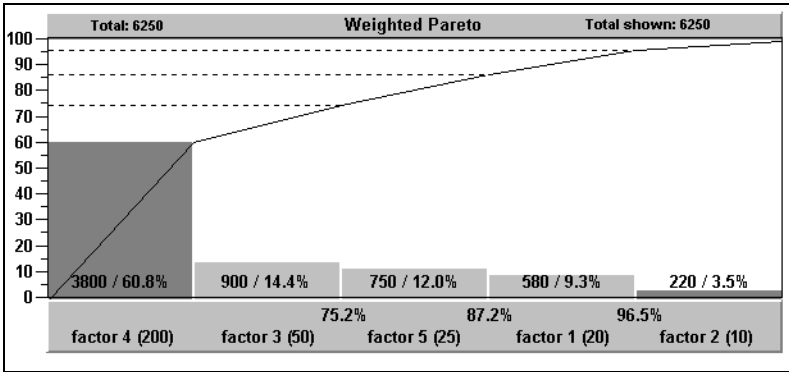
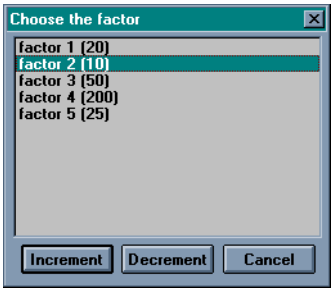


Figure 18-88. Weighted Pareto Chart

Incrementing Factor Counts

You can use a Pareto chart to interactively or automatically increment factor counts. In fact, the object can accept any combination of automatic and interactive inputs.

If you select the **Allow Interactive Choice** option in the object definition dialog box, the operator can manually increment factor counts. To do this, the operator clicks the mouse cursor on the Pareto chart object. This pops up box that lists all the possible factors you can choose. The factor that the cursor is on when the operator clicks on the chart determines which factor is highlighted when the dialog box pops up. In this example, the mouse cursor was over the factor 2 bin.



If you select the **Allow Decrement Count** option in the object definition dialog box, the Decrement button is displayed in the pop-up, allowing the operator to decrement a factor count. Operators with a high enough security level can correct errors if necessary.

Pareto Data Members

The Pareto object can use external connections to trigger the factor counts automatically. You can connect a pushbutton, an alarm, an expression, a PLC output, and so on, to a factor trigger to increment a factor count on a transition from FALSE to TRUE. The following table lists data members of the Pareto object class.

Table 18-84. Pareto Data Members

Data Members	Type	Read	Write	Description
visible	logical	yes	yes	When TRUE, the Pareto chart becomes visible on the control panel. When FALSE, it is invisible. The default value is TRUE.
total	numeric	yes	no	The total of all unweighted factor counts in all defined factors.
total.shown	numeric	yes	no	The total of all unweighted factor counts in all currently displayed factors.
total.weighted	numeric	yes	no	The total of all weighted factor counts in all defined factors.
total.shown.weighted	numeric	yes	no	The total of all weighted factor counts in all currently displayed factors.
factor1 – 1000	logical	no	yes	On transition from FALSE to TRUE, increment the respective factor count by 1.
count1 – 1000	numeric	yes	no	The count in the respective unsorted factor bin.
count.sorted1 – 1000	numeric	yes	no	The count in the respective sorted-by-count factor bin.
count.weighted1 – 1000	numeric	yes	no	The weighted count in the respective unsorted factor bin.
count.weighted.sorted 1 – 1000	numeric	yes	no	The weighted count in the respective sorted-by-weighted-count factor bin.
label1 – 1000	text	yes	no	The label of the respective unsorted factor bin.
label.sorted1 – 1000	text	yes	no	The label of the respective sorted-by-count factor bin.
label.weighted.sorted 1 – 1000	text	yes	no	The label of the respective sorted-by-weighted-count factor bin.
percent1 – 1000	numeric	yes	no	The percent of the total count in the respective unsorted factor bin.

Table 18-84. Pareto Data Members (Continued)

Data Members	Type	Read	Write	Description
percent.sorted1 – 1000	numeric	yes	no	The percent of the total count in the respective sorted-by-count factor bin.
percent.weighted 1 – 1000	numeric	yes	no	The percent of the total weighted count in the respective unsorted factor bin.
percent.weighted.sorted 1 – 1000	numeric	yes	no	The percent of the total weighted count in the respective sorted-by-weighted-count factor bin.

Philips

Philips is a protocol driver class Lookout uses to communicate with Philips devices using the PPCCOMM communication protocol.

A Philips object contains a great deal of data. It supports reading and writing of all predefined data points. When you create a Philips object, you have immediate access to all the object data members (see *data member* list below).

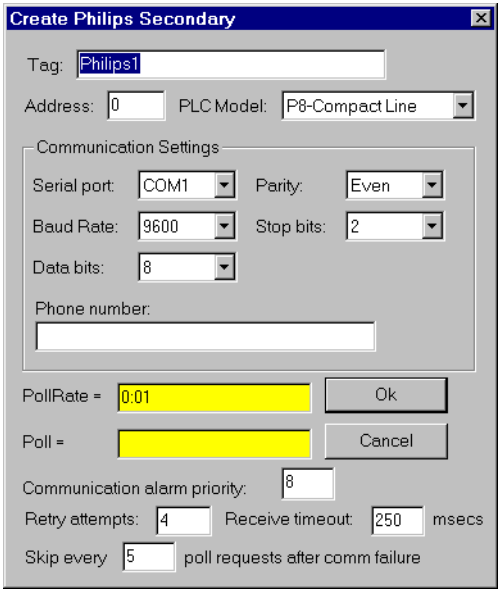


Figure 18-89. Philips Configuration Parameters Dialog Box

Serial port specifies which communications port the object uses for communicating to the external device. This does not specify the communication type. Communication type is determined by the **Options»Serial Ports…** command.

Baud rate indicates the baud rate that Lookout uses to communicate with the hardware device. This setting should match the selection made on the physical device.

Data bits indicates the number of data bits that Lookout uses to communicate with the hardware device. This setting should match the selection made on the physical device.

Stop bits indicates the number of stop bits that Lookout uses to communicate with the hardware device. This setting should match the selection made on the physical device.

Parity indicates the parity that Lookout uses to communicate with the hardware device. This setting should match the selection made on the physical device.

Phone number specifies the number to be dialed if the selected serial port is configured for dial-up. This number only applies to the individual protocol object.

PollRate is a numeric expression that determines how often to poll the device. The object then polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *Introduction*, for information on entering time constants.

Poll is a logical expression. When this expression changes from FALSE to TRUE, Lookout polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Communication alarm priority determines the priority level of alarms generated by the Philips object. Such alarms are typically related to communications with the physical device.

Retry attempts specifies the consecutive number of times Lookout attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the Philips object generates an alarm and releases the communication port back to the communications service which then moves on to the next device in the polling queue (if any). Refer to Chapter 6, *Serial Communications*, for more information.

Receive timeout is the time delay Lookout uses in waiting for a response from a device before retrying the request.

The **Skip every…** setting instructs Lookout not to poll a device it has lost communication with on every scheduled poll. Instead, Lookout skips the device in the polling cycle accordingly. Once communications have been reestablished, the device is polled on its regular cycle.

Philips Data Members

As with all Lookout drivers, you can access I/O points and other data through data members. The following is a table of data members currently supported by the Philips object class.

Table 18-85. Philips Data Members

Data Member	Type	Read	Write	Description
IX0.0 – IX123.15	logical	yes	no	Discrete input, 1 bit
IB0.0 – IB1023.1	numeric	yes	yes	Discrete inputs, 8 bits
IW0 – IW1023	numeric	yes	yes	Discrete inputs, 16 bits
ID0 – ID1023	numeric	yes	yes	Discrete inputs, 32 bits
MX0.0 – MX14335.15	logical	yes	no	Data register, 1 bit
MB0.0 – M14335.1	numeric	yes	yes	Data register, 8 bits
MW0 – MW14335	numeric	yes	yes	Data register, 16 bits
MD0 – MD14335	numeric	yes	yes	Data register, 32 bits
QX0.0 – QX1023.15	logical	yes	no	Discrete output, 1 bit
QB0.0 – QB1023.1	numeric	yes	yes	Discrete outputs, 8 bits
QW0 – QW1023	numeric	yes	yes	Discrete outputs, 16 bits
QD0 – QD1023	numeric	yes	yes	Discrete outputs, 32 bits
Poll	logical	no	yes	When this value transitions from FALSE to TRUE, Lookout polls the device.
PollRate	numeric	no	yes	Lookout expression that determines the device polling frequency.
Update	logical	yes	no	Object-generated signal that pulses low each time it polls the device.
CommFail	logical	yes	no	Object-generated signal that is on if Lookout cannot communicate with the device(s)

Philips Status Messages

Invalid frame format
Bad Checksum
Invalid frame length
These messages indicate garbled communications. Check cabling for signal noise or multiple devices with the same address.

Philips Alarms

These are messages returned to Lookout in a response frame. See your Philips documentation for meanings and solutions

PID

The PID object compares a **Process Variable** to a **Setpoint**. If there is a difference, it calculates the error and adjusts its output to compensate until the **Process Variable** is equal to the **Setpoint**.

PID stands for Proportional-Integral-Derivative. These are three factors in the equation that can be applied against the calculated error. You specify **Gain** which is the proportional factor, **Reset** (the integral factor), and **Rate** (the derivative factor) to define how the object responds to the error.

☞ **Note** ***The way in which the PID object responds to your process can vary greatly according to the parameters you enter and the process you are controlling. Any discussion regarding tuning of a PID loop falls outside of the topics addressed in this manual.***

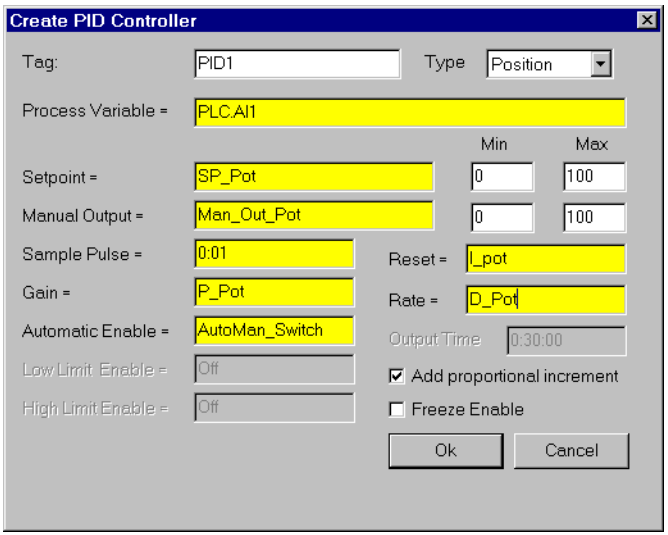


Figure 18-90. PID Definition Parameters Dialog Box

Type selects either positional control or velocity control. Detailed descriptions of both control modes are provided below.

Process Variable (*PV*) is typically the numeric signal from the field that you want to control. The PID loop equation does not expect this value to be normalized; rather the PID object performs the scaling of loop input and output values from engineering units.

Setpoint (*SP*) is typically the value of a Pot object, a constant numeric value, or the output signal from another PID object in a cascaded loop. Like the process variable, the setpoint is also scaled internally by the PID object.

Setpoint Min and **Max** are numeric constants that specify the range of *SP* and *PV* in engineering units.

Manual Output is a numeric parameter that specifies the output of the object when it is in manual mode; that is, when the **Automatic Enable** expression is FALSE. Users typically enter either a constant, or the name of a Pot object in this field.

Output Min and **Max** are numeric constants that specify the range of the object output signal. The output is often referred to as the *manipulated variable* (*MV*).

Sample Pulse indicates the frequency at which the PID object executes. This parameter field can contain either a numeric constant or a logical variable. If you use a numeric constant (like `0:01` for one second), the object calculates a new output value at the defined frequency. If you use a logical variable, then the variable should pulse at some desired frequency. Any time the pulse transitions from FALSE to TRUE, the object calculates a new output value. It is very important not to over-sample your data. Start with a slow sample rate.

Gain (K_c) is a numeric parameter that determines the overall sensitivity of the PID loop to changes in error. A gain value of 1.00 changes the proportional increment of the PID equation by 50 percent when there is a 50 percent change in error. A gain value of 0.25 changes the proportional increment by 12.5 percent with a 50 percent change in error.

Reset (T_i) also referred to as integral time, is a numeric parameter that specifies the amount of time it takes for the integral sum increment of the PID loop equation to react to a give change in error. For example, if the error suddenly changes by 20% and reset = 0:10 (10 seconds), the integral increment of the PID loop increases at a rate of 0.5 percent per second until it has changed by 20 percent after 40 seconds. This 20 percent contribution is multiplied by the gain, so if the gain is 2.0, the integral term contributes 40% to the loop output in this example. In other words, the shorter the reset time, the faster the object output responds to a change in either *PV* or *SP*.

Rate (T_d) also referred to as derivative time, is a numeric parameter that dampens loop response. It is calculated based upon the rate of change of *PV* and adds an increment to the output that attempts to anticipate and slow the change in *PV*. As an example, if *PV* is increasing by 10 percent per minute,

and *rate* is 0:30 (0.5 minutes), the derivative increment is calculated as –(10%/min. × 0.5 minutes)= –5% (or –0.05). So the derivative term would contribute –5% to the output if the gain is 1.0.

Automatic Enable specifies whether the loop controller is operating in automatic mode or manual mode. When it is ON, controller is operating in automatic mode and the output signal is being calculated using the PID algorithm. In manual mode, the output signal is equal to the **Manual Output** input signal.

The PID object provides bumpless transfer from manual to automatic operation—when the controller is switched from manual to automatic, its output begins changing from the current manual output setting. Contrast this with a loop controller without bumpless transfer. When such a controller is in manual, the integral term continues to accumulate. When the controller is switched to automatic, the loop controller would immediately go to a high or low output.

Add proportional increment specifies whether the loop equation adds the proportional increment of the PID equation to *MV*. This value is typically ON.

Freeze Enable specifies whether the loop bias should be frozen or actively back-calculated when the controller output signal goes out of range. In either case, the loop controller is protected from integral wind-up, but if Freeze Enable is OFF (recommended setting), the bias is actively back-calculated to prevent controller overshoot when *PV* comes back into range.

The PID object protects against integral windup in one of two selectable ways: It either freezes the bias term when the controller output goes out of range, or it actively back-calculates the bias so the controller responds smoothly with less chance of overshoot when its output returns to range.

Output Time is a numeric constant that specifies the time domain of the controller output when operating in velocity mode. For example, if the object output controls a value with dimensions of inches per minute, output time would be 1:00 (one minute).

Low Limit Enable is used in velocity control mode. It is an optional logical signal that clips the PID output to a value grater than or equal to zero when TRUE. This input can be used to signal the controller that the low limit switch on the controlled device has been activated.

High Limit Enable is used in velocity control mode. It is an optional logical signal that clips the PID output to a value less than or equal to zero when TRUE. This input can be used to signal the controller that the high limit switch on the controlled device has been activated.

PID Positional Control

When you select `Position`, the PID object calculates the output as follows:

$$MV = K_c (e_n + \text{integral sum}_n - T_d / dT (Pv_n - Pv_{n-1}))$$

where:

dT = time increment between current and previous calc.

MV = controller output (manipulated variable)

K_c = controller gain (units: % output / % error)

e_n = error at sample n (error = $SP - Pv_n$)

$\text{integral sum}_n = \text{integral sum}_{n-1} + dT / T_i (e_n - e_{n-1})$ also called bias

T_d = rate, or derivative time

$Pv_n - Pv_{n-1}$ = change in PV from previous to current calculation

PID Velocity Control

The output of the velocity form of the PID equation is the velocity or rate of change of the output signal. The velocity form of the PID equation is the first derivative of the position form of the PID equation with respect to time, so the result is the rate of change of the controller position.

When you select `Velocity`, the object calculates the output as follows:

$$dMV = MV_n - MV_{n-1}$$
$$= K_c ((e_n - e_{n-1}) + T_s e_n / T_i + T_d/T_s (Pv_n - 2PV_{n-1} + PV_{n-2}))$$

where:

MV = controller output position (manipulated variable)

dMV = contoller output velocity

K_c = controller gain (units: % output / % error)

e_n = error at sample n (error = SP – Pv_n)

T_d = rate, or derivative time

PV = process variable

PID Data Members

Table 18-86. PID Data Members

Data Members	Type	Read	Write	Description
Error	numeric	yes	no	This is the difference between the **Setpoint *(SP)*** and the **Process Variable *(PV)*.**
Output	numeric	yes	no	This is the PID controller output value, also called *MV*, or the manipulated variable. This value is either the result of the PID loop equation or the **Manual Output**, depending on the state of **Automatic Enable**.

Comments ***The proportional term of the PID equation contributes an amount to the output equal to the error multiplied by the Gain. This provides an immediate output compensation when the error value changes.***

The integral term of the PID equation calculates a running total of the error summed (or integrated) over time—think of this increment as adding the area under the curve of a plot of error versus time. While *SP* is greater than *PV*, the integral term is increasing, and while *PV* is greater than *SP*, the integral term is decreasing. The sensitivity of the integral output is set by the gain and the reset variables. Integral action can be eliminated by setting **Reset** to a higher number. At least some Integral action is required, however, for the loop controller to operate properly with bias adjustment. If you do not use any Integral, you may experience offset, a condition in which the output is adjusted to compensate for the error, but not enough to correct the error.

The derivative term of the PID equation acts to dampen the change in *PV* by adding a negative value for a positive-going *PV* and a positive value for a negative-going *PV*. Because *PV* is subject to sudden small changes and signal noise in many process loops, derivative action can cause a loop to respond erratically. **Rate** can be set to 0, especially when initially tuning

the loop, to eliminate derivative action. Derivative action dampens process loops that tend to oscillate around the setpoint and thus provide better loop response. Rapidly changing loops such as liquid flow control in a pipe may not benefit from derivative action, but more sluggish loops that tend to build momentum, such as temperature control, benefit from derivative action by preventing overshoot and dampening oscillatory action.

Pipe

Pipe displays different color rectangles (pipes) on a control panel as defined by the values of **Conditional expressions**. Pipe determines which color rectangle to display based on the order and current status of your **Conditional expressions**. If several **Conditional expressions** are TRUE at once, Pipe displays the color associated with the first TRUE expression. **Conditional expressions** must result in logical values.

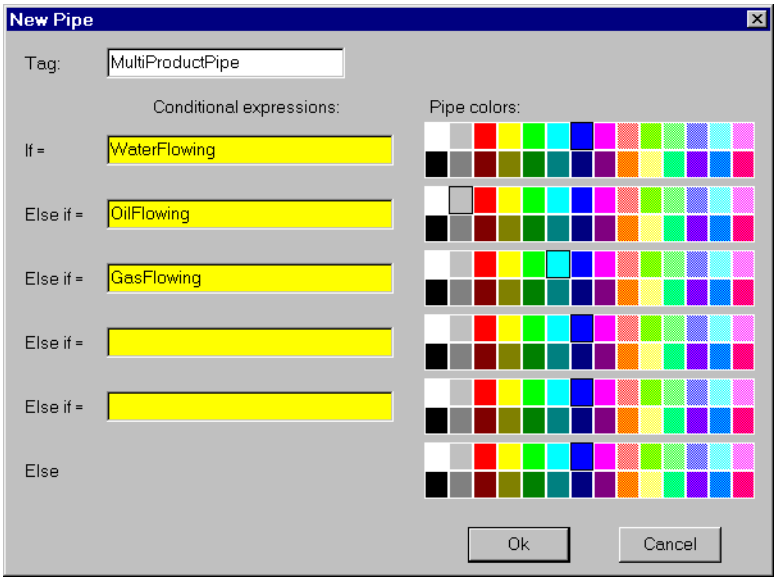


Figure 18-91. Pipe Definition Parameters Dialog Box

Pipe Data Members

Table 18-87. Pipe Data Members

Data Members	Type	Read	Write	Description
none	—	—	—	Pipe does not have data members.

Comments ***You can easily create a complex piping network scheme (including changing colors) with a single Pipe object. Display the object on a control panel and copy the pipe display with the shift-drag method to create additional pipes with the same parameters. You can then move, resize, and group the pipes as you choose.***

Playwave

Playwave connects Microsoft standard wave form files (`.WAV`) to events in Lookout. Playwave plays the audio file specified by **Wave file** when **Play when** transitions from off to on. **Play when** is a logical expression and might range from a simple pushbutton, to a digital input from a PLC, to an alarm generated in Lookout. You can also create your own custom audio files with various software products. Therefore, you can connect individual alarms to custom wave files to be played each time an alarm goes TRUE.

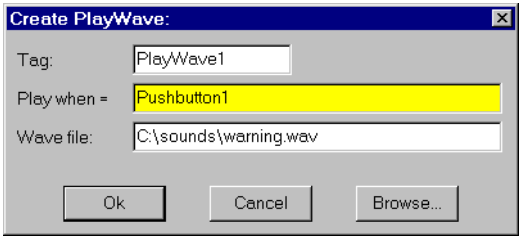


Figure 18-92. Playwave Definitions Parameters Dialog Box

Playwave Data Members

Table 18-88. Playwave Data Members

Data Members	Type	Read	Write	Description
none	—	—	—	Playwave does not have any data members

Comments ***Many computers do not come equipped with quality speakers built in. If this is the case, your wave files may sound distorted or may even be inaudible. If you want to take advantage of the Playwave feature, you may need to buy additional hardware, in particular a Microsoft Windows compatible sound board (with Windows driver) and external speakers.***

Pot

Pot is a potentiometer that you use to change numeric setpoint values. You can display pots on a control panel as a knob, vertical slider, horizontal slider, increment/decrement pushbuttons, or digital entry. You can also use pots as multiple-position switches.

If you change the background color of a panel and add a pot object displayed as a slider, its color is always gray. To change the background color of a pot to match your panel, select the pot object, then pick **Change»Background Color** from the menu.

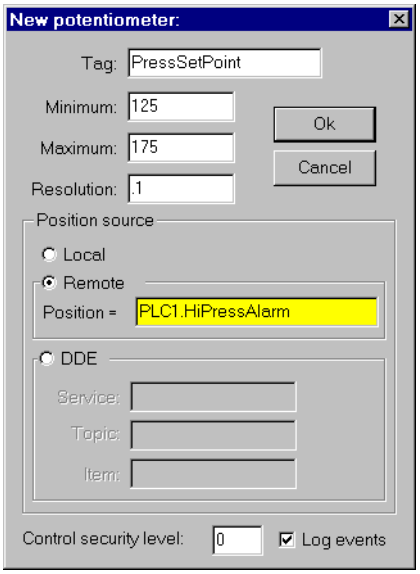


Figure 18-93. Pot Definitions Parameters Dialog Box

Minimum is the lowest value signal the pot will generate.

Maximum is the highest value signal the pot will generate.

Resolution is the smallest increment of change, or detent spacing the pot supports.

Position source determines where the value of the pot resides. **Local** indicates the value of the pot lies within the object itself—on the control panel.

Remote pots get their values from a remote source, often the register on a controller they are connected to. Adjusting the pot changes the value in the register, and changing the value in the register adjusts the pot. In effect, the pot is tracking a remote value. This is especially useful when you want to prevent Lookout from changing the value of setpoints or registers upon initial startup, or reconnection of lost communication. When you use this style pot you are creating a kind of looped signal. Half the loop is formed when you connect the controller register to the pot with the **Position** expression, while the second half is formed when you connect the pot output signal to the controller register. **Position** is a numeric expression. Do not forget to complete the second half of the loop with the **Object»Edit Connections…** command.

Much like Remote pots, **DDE** (Dynamic Data Exchange) pots get their values from a remote source. This could be a cell in a spreadsheet, another DDE aware application, or a second copy of Lookout running on the network. The last DDE parameters used on any object automatically become the default values for any new DDE object. See Chapter 13, *Developer Tour*, for more information on Service, Topic, and Item parameters.

Control security level specifies the minimum security level operators must have to gain access to this individual object, and thus control it.

The **Log events** option creates a permanent audit trail for the object—who did what and when. All adjustments of the pot are logged to disk, including the time of the adjustment, the operator account name, and what adjustment was made. See Chapter 11, *Logging Data and Events*, for more information on event logging.

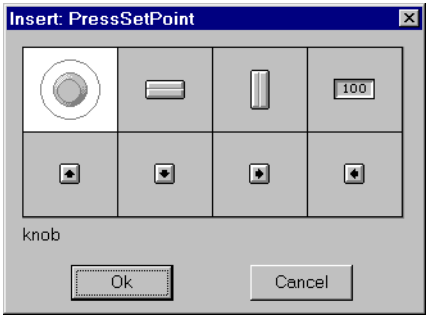


Figure 18-94. Pot Display Parameters Dialog Box

☞ **Note** ***You can modify the background color on vertical and horizontal sliders with the* Change»Background color… *menu command. You can modify the font and font color of digital pots using* Change *commands.***

Pot Data Members

Table 18-89. Pot Data Members

Data Member	Type	Read	Write	Description
(implicit)	numeric	yes	no	Current value
decrement	logical	no	yes	When this data member value transitions from FALSE to TRUE, the implicit value of the pot object decreases by the pot **Resolution** amount.
enable	logical	no	yes	If TRUE (the default), enables DDE. If FALSE, disables DDE. The default value is on. This input is ignored for non-DDE TextEntry objects.
increment	logical	no	yes	When this data member value transitions from FALSE to TRUE, the implicit value of the pot object increases by the **Resolution** amount.
visible	logical	no	yes	When FALSE, the pot object cannot be seen on the display panel. When TRUE, the pot can be seen and controlled.

Comments ***Potentiometers are one of the most common control objects used in process controls. Using pots a plant operator can make setpoint changes with the mouse. Pots also work well as H-O-A switches. To create an HOA switch with a pot, specify the minimum as 1, the maximum as 3, and the resolution as 1. See the* Developer Tour *for example of a Pot used as a HOA switch.***

The increment and decrement data members enable quick connection of pot objects to $Keyboard and Panel function keys, and screen Pushbuttons. These are often used to control pot objects when Lookout is running on an industrial PC platform that has restricted or no mouse functionality.

Profibus DP

Profibus DP is a protocol driver class for communicating with PLCs and remote I/O units using the Decentralized Periphery protocol in the Profibus standard. The Lookout driver currently supports the S-S Technologies card 5136-PFB PC card only.

Configuring the Profibus DP Network

In order to use the Lookout Profibus DP driver for the S-S card, you must first configure your DP network using the Siemens COM ET-200 software. This software is not included in the Lookout release and must be purchased separately. The S-S card is the master device on the Profibus network. Because the COM ET-200 software does not know about this device, you must select some other device as the master. You may leave all of its parameters as the defaults because the S-S card ignores the parameters of this master system.

You must, however, enter a master station number and configure the bus parameters. These parameters are used by the SS card to configure the DP network.

Next, you must configure the parameters for each of the slave devices. Refer to the online documentation for the COM ET200 software for instructions on how to do this. Once all the slave devices are configured, save the file to an `.et2` format and export the file to a binary file `.2bf` format. This last step is done using the Export selection in the File menu. Place the binary file in a location accessible to Lookout.

Profibus DP Requirements

To run the 16-bit version of this object, you must be running Windows 3.1 or Windows 95, with an S-S Technologies 5136-PFB card installed in the computer. Pre-install the firmware module `pfbprofi.ss1` on the card (using the `pfbinst` utility). The usual way to do this is to insert a call to `pfbinst` in your `autoexec.bat` file.

To run the 32-bit version of this object, you must be running Windows NT 3.51 or greater or Windows 95, with an S-S Technologies 5136-PFB card installed in the computer. Lookout automatically loads the firmware module when an object is created.

Both versions require that the configuration file in COM ET 200 binary format be placed in a directory accessible to Lookout.

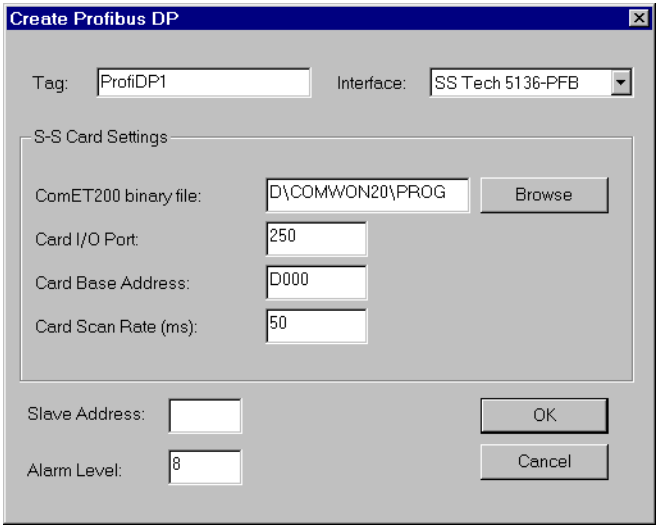


Figure 18-95. Profibus DP Configuration Parameters Dialog Box

Interface is the type of card to be used to communicate with the DP network. At present, this driver only supports the S-S card interface.

Slave Address is the Profibus address for the particular slave device that this object is to communicate with.

Alarm Level determines the priority level of object-generated alarms (0 – 10).

PFB Card Settings

COM ET200 Binary File is the path to the binary configuration file produced by the Siemens COM ET200 software. This file must match the network that the S-S card is actually connected to.

Card I/O Port specifies the base port address for the card. The jumpers on the card must be preset to the port address selected.

Card Base Address specifies the base address location of the card memory. At present, only one 5136-PFB card in a computer is supported. The default is D000. In 16-bit Lookout, this value is preset when loading the firmware onto the card. In 32-bit Lookout, the Profibus DP object itself sets this on the card when it is loading the firmware.

Network Init Timeout specifies the number of milliseconds the card should wait while attempting to initialize the network before generating an alarm. The default is 1000 ms or one second.

Card Scan Rate is the rate at which the 5136-PFB card is scanned to look for fresh data. The default is 50 milliseconds.

Profibus DP Data Members

All readable and writable members—inputs/outputs—are bundled with the object. Therefore, as soon as you create a Profibus DP object you immediately have access to all the object data members.

Data can be addressed either as bytes or words within slots, or as bits within these bytes or words. However, you cannot address a slot configured for digital input or output (as bytes) using word numbers. Similarly you cannot address a slot configured for analog input or output (as words) using byte numbers.

Table 18-90. Profibus DP Data Members

Data Member	Type	Read	Write	Description
CommFail	logical	yes	no	Object-generated signal that is on if, for any reason, Lookout cannot communicate with the Profibus card.
Slot0B0 – Slot127B63	numeric	yes	yes	This addresses data in a slot as a sequence of bytes.
Slot0B0.0 – Slot127B63.7	logical	yes	yes	This addresses individual bits while looking at a slot as a sequence of bytes.
Slot0W0 – Slot127W63	numeric	yes	yes	This addresses data in a slot as a sequence of words.
Slot0W0.0 – Slot127W63.15	logical	yes	yes	This addresses individual bits while looking at a slot as a sequence of words.

Profibus DP Status Messages

Profibus SS Card not found
The SS card was not found at the specified port and card base address. Check that the jumper settings on your card match the port address that you specified in the object creation dialog box. Also, check that the card is

properly seated in the slot. In the 16-bit version, make sure you are loading the `pfbrofi.ss1` module onto the card prior to starting Lookout.

Profibus init failed
The initialization of the Profibus SS card failed. This alarm is usually followed by an explanation of the reason for this failure.

Error accessing COM ET 200 binary file
A file error occurred while attempting to open the COM ET 200 binary file.

Unconfigured Read Error
A read was attempted on a data member that is not valid in the current Profibus DP network configuration.

Unconfigured Write Error
A write was attempted on a data member that is not valid in the current Profibus DP network configuration.

Slave Error
There was a problem with the particular slave device you are trying to access. This alarm is followed by a description of the problem with that slave device.

ProfibusL2

ProfibusL2 is a protocol driver class for communicating with PLCs using Profibus Sinec-L2.

One of the problems with communicating with PLCs using Profibus Sinec-L2 is that there is no standard messaging system for transferring data. The Sinec L2 interface has a variety of message types available that you can use with applications as well as PLCs on a network. Lookout uses its own messaging system for transferring data and provides help for you to implement the PLC end of the messaging system.

Lookout Messaging System

You should read the Siemens manual *Sinec-L2 Interface of the S5-95U Programmable Controller*, or some similar document that details a Sinec-L2 interface and how to write programs for your particular Sinec-L2 interface device.

Lookout uses two message types to receive and send all the data it needs. It uses the message type SRDh (Send and Request Data, high priority) to receive data from the PLC. To send data to the PLC, Lookout uses the message type SDAh (Send Data with Acknowledge, high priority).

All messages in Sinec-L2 are directed at specified SAPs (Service Access Points). You must program SAPs on each PLC to provide or accept the data you want to exchange with Lookout. You can use a given SAP only for reading or writing. Each poll request for a configured SAP reads *all* the data in that SAP. In the same way, if you are writing to a particular SAP, each write transfers all the data for that SAP to the PLC.

Sample Program

Included in your copy of Lookout is a sample Siemens STEP5 program file, `LKTPFBST.S5D`. This file is placed in the `\sinecl2` directory under your Lookout directory during installation. This program contains an example of the logic necessary to create the Service Access Points (SAPs) for communications from a Simatic S5-95U to Lookout software using the Profibus protocol.

The program consists of one program block, three function blocks, three data blocks, and the modifications needed in OB1 and DB1 to successfully configure the Profibus communication link. This sample establishes two SAPs to the Profibus Layer 2 services of the PLC. Lookout uses the first

SAP, 34, to write values to the PLC. Lookout uses the second SAP, 35, when reading values from the PLC. You can read or write data in byte, word, or double word formats.

In order to install the example Profibus program, follow the steps below:

1. **TRANSFER ALL BLOCKS TO THE CPU MEMORY**

   This must include all PBs, FBs, and DBs.

2. **PROFIBUS CONFIGURATION**

   The example configures SAP 34 and SAP 35. However, you may modify the program, if necessary, to customize your application.

3. **PROFIBUS EXECUTION**

   Jump to PB 223 is required from OB1. An example of this jump is included in OB1 in the program.

4. **PROFIBUS DATA HANDLING**

   The sample program uses DB 234, DB 235, and DB 236 to move data from the PLC to Lookout and from Lookout to the PLC. All data must be mapped to and from data blocks for Profibus communications. These data blocks also contain 8-byte headers used as a part of the Profibus configuration parameters. Again, you may modify these blocks if necessary. In addition, flag bytes above 200 are used by the example program, so you should avoid using them.

Detailed Explanation of the Profibus Example Program

The example program employs the Layer 2 services of the Siemens Simatic S5-95U for the Profibus communications to Lookout. In particular, the program uses SRD and RUP_MULTIPLE. The configuration of these services is described in detail in Chapter 8, *Data Transmission by Accessing Layer 2 Services*, of the *Sinec-L2 Interface of the S5-95U Programmable Controller manual.*

DB1 Configuration

The first step in configuring the S5-95U is to insert the correct Profibus parameters in DB1 of the PLC. You should employ the COM_DB1 software, available from Siemens, when setting these parameters.

First, you must enter the SINEC L2 basic parameters. These parameters include the station address, baud rate, target rotation time, and more. The parameters must exactly match those used in configuring your SinecL2 object. You can find the defaults for these parameters in the example program where they are used.

Next you must configure the SINEC L2 layer 2 service. Here, you define the individual SAPs and designate a status byte for each SAP. SAPs used for writing data to the PLC must have a status byte allocated in the **Receive** area. SAPs used for reading data from the PLC must have a status byte allocated in both the **Send** and **Receive** areas.

You can define up to 23 SAPs, numbering from 33 to 54 and also including 64. The example program uses SAP 34 for writing data, and FW 204 as its status and length bytes. The program uses SAP 35 for reading data, with FW 200 and FW 202 used for the status and length bytes in the **Send** and **Receive** areas.

Function Block Explanation

The example program contains three function blocks to handle the data transfer between the Profibus SAPs and the CPU internal memory. These blocks evaluate the status bits associated with each SAP, and send or receive the data when allowable.

Function Block FB 224 coordinates the data written from Lookout to the PLC. The CPU looks for an *indication* from the SAP that new data exists. If the receive is viable, the block performs a jump to FB 253, which is the integrated L2-REC function block. The block takes the data and writes it to the area specified by DBNR. The example uses DB 234, starting at DW 0 and continuing for an unknown wildcard length. The first 8 bytes of DB 234, DW 0 through DW 3, contain the header for the message. You must include this header in the data block for proper operation.

Function Blocks FB 223 and FB 225 handle the data Lookout is attempting to read from the PLC. FB 225 acts in the same way as FB 224, looking for an indication that the SAP has received a new request for data. FB 223 actually writes the data to the SAP using a jump to FB 252, the integrated L2-SEND function block. The data is taken from the area designated by DBNR. In this case, DB 235 is used by FB 223 and DB 236 is used by FB 225. The first 8 bytes act as a header to the message to follow, and must be set with specific parameters. In DB 235, DW 1 through DW 4 are used for the Send, and DW 128 through DW 131 are used for the Indication. In DB 236, DW 1 through DW 4 are used for the Indication. The parameters required in the data blocks are discussed in the *Sinec-L2 Interface of the S5-95U Programmable Controller manual*.

Program Block PB 223 initiates the jumps to all the function blocks for the Profibus communications. You must add a jump to PB 223 in your OB 1 in order to start the communications.

Finally, all the program blocks and function blocks use bits of the **Send** and **Receive** status bytes in their logic. If you change these bytes in DB 1, the you must also change the associated logic accordingly.

ProfibusL2 Requirements

To run the 16-bit version of this object, you must be running Windows 3.1 or Windows 95 and have an S-S Technologies 5136-PFB card installed in the computer. You must pre-install the firmware module `pfbprofi.ss1` on the card (using the `pfbinst` utility). The usual way to do this is to insert a call to `pfbinst` in your `autoexec.bat` file

To run the 32-bit version of this object, you must have Windows NT 3.51 or better, or Windows 95 loaded, and an S-S Technologies 5136-PFB card installed in the computer. Lookout automatically loads the firmware module when an object is created.

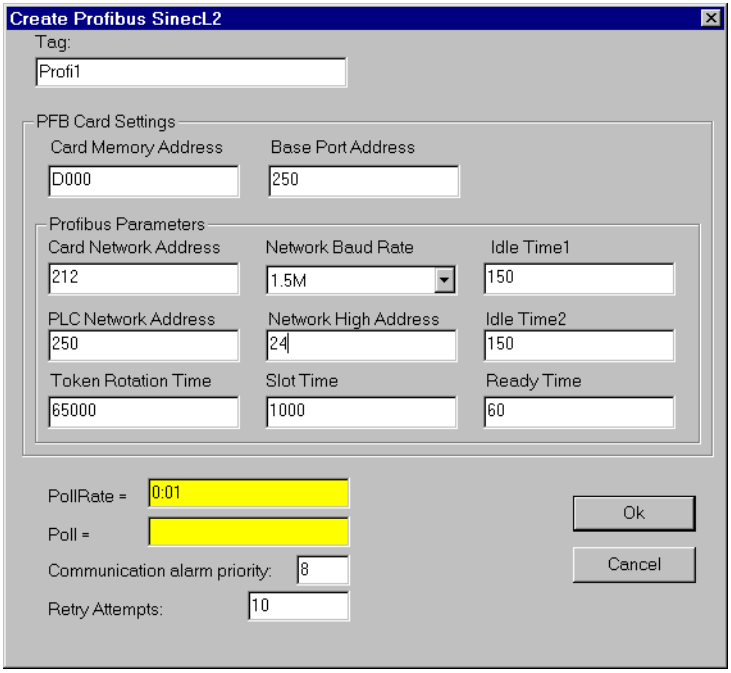


Figure 18-96. ProfibusL2 Configuration Parameters Dialog Box

PollRate is a numeric expression that determines how often to poll the device. Lookout then polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See

Numeric Data Members in Chapter 2, *Introduction*, for further information on entering time constants.

Poll is a logical expression. When this expression changes from FALSE to TRUE, Lookout polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Communication alarm priority determines the priority level of object-generated alarms (0 – 10).

Retry attempts specifies the consecutive number of times Lookout attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the Profibus L2 object generates an alarm.

Card Memory Address specifies the base address location of the card memory. At present, only one 5136-PFB card in a computer is supported. The default is D000. In 16-bit Lookout, this value is preset when loading the firmware onto the card while in 32-bit Lookout, the Profibus object itself sets this on the card when it is loading the firmware.

Base Port Address specifies the base port address for the card. The jumpers on the card must be preset to the port address selected.

PFB Card Settings

Card Network Address specifies the Profibus address for the card on the bus. The valid range is 0 to 126.

PLC Network Address specifies the Profibus address for the PLC on the bus. The valid range is 0 to 126.

Network High Address specifies the highest possible Profibus address possible on the bus. The valid range is 0 to 126.

Token Rotation Time specifies the target maximum token rotation time for the network in Tbits (bit times). The valid range is 256 to 16,777,215.

Network Baud Rate specifies the baud rate to be used on the network. The valid selections are shown in the selection box.

Slot Time specifies how long the card waits for a reply to a message, in Tbits. The valid range is 37 to 16,383.

Idle Time1 specifies the time in Tbits that the card waits after it receives a reply, an acknowledge or a token message before sending a message. Range: 35 to 1023

Idle Time2 specifies the time in Tbits that the card waits after sending an SDN (Send Data with no acknowledge) message before it sends again. Range: 35 to 1023

Ready Time specifies the time in Tbits that the card, after it sends a command, is ready to receive the ACK or response. It is also the time the card waits after receiving a command. The valid values range from 11 to 1023.

ProfibusL2 Data Members

All readable and writable members (inputs/outputs), polling instructions, and so on, are bundled with the object. Therefore, as soon as you create a Profibus object you immediately have access to all the object data members (see data member list below).

Data is addressed using SAP numbers and, within a SAP, the byte, word or Dword number. For example, if SAP 33 is configured for reading, SAP33B200 refers to the 201st byte of that SAP. Similarly, SAP42DW10.20 refers to the 21st bit of the 11th Dword of SAP 42.

The legal SAPs used by the Lookout Profil2 driver range from 0 to 255. Each SAP may provide up to 242 bytes of data (0 – 241) which you can address either as bytes, 121 words, or 60 Dwords. The actual range of valid SAPs varies according to the device being addressed. You can address bits by appending a period and bit number after the byte number, word number, or Dword number.

☞ **Note** ***Lookout protocol driver objects automatically generate an efficient read/write blocking scheme based on the inputs and outputs being used in your process file. You are not required to build your own I/O blocking table.***

Table 18-91. ProfibusL2 Data Members

Data Member	Type	Read	Write	Description
CommFail	logical	yes	no	Object-generated signal that is on if, for any reason, Lookout cannot communicate with the Profibus card.
Poll	logical	no	yes	When this value transitions from FALSE to TRUE Lookout polls the device.
PollRate	numeric	no	yes	Lookout expression that determines the device polling frequency.
SAP0B0 – SAP255B241	numeric	yes	yes	This addresses all the data in the SAP as a sequence of bytes.
SAP0B0.0 – SAP255B241.7	logical	yes	yes	This addresses individual bits while looking at an SAP as a sequence of bytes.
SAP0W0 – SAP255W120	numeric	yes	yes	This addresses all the data in the SAP as a sequence of words.
SAP0W0.0 – SAP0255W120.15	logical	yes	yes	This addresses individual bits while looking at an SAP as a sequence of words.
SAP0DW0 – SAP255DW60	numeric	yes	yes	This addresses all the data in the SAP as a sequence of double words.
SAP0DW0.0 – SAP255DW60.31	logical	yes	yes	This addresses individual bits while looking at an SAP as a sequence of double words.
Update	logical	yes	no	Object-generated signal that pulses low each time it polls the device.

ProfibusL2 Status Messages

Profibus SS Card not found

The SS card was not found at the specified port and card base address. Check that the jumper settings on your card match the port address that you specified in the object creation dialog box. Also, check that the card is properly seated in the slot. In the 16-bit version, make sure you are loading the `pfbrofi.ss1` module onto the card prior to starting Lookout.

Profibus SS Card timed out on message
The SS card did not respond within a reasonable time to the message sent by Lookout. This may mean that you have lost communication with the card. Try restarting Lookout to see if this fixes the problem (reloading the firmware if you are using 16-bit Lookout). If the problem persists, call National Instruments for assistance.

Pulse

Pulse is a timer that generates a periodic pulse of a specified duration. When **On/off signal** transitions to ON, the output signal turns on for the pulse duration time and then turns off for the remainder of the period. The output signal immediately turns off when the **On/Off signal** goes low.

Timer period is the time interval for the full pulse cycle, and **Timer duration** is the width of each pulse. These parameters can range from 0.0 seconds to several years, with an effective resolution of 0.1 seconds over the entire range. **Timer duration** should always be less than **Timer period**.

The object is represented on a control panel by showing the time remaining before the output changes state. It is depicted in the format defined by the **Display format** parameter. It is updated approximately once per second. If the **On/Off signal** is FALSE, it shows OFF.

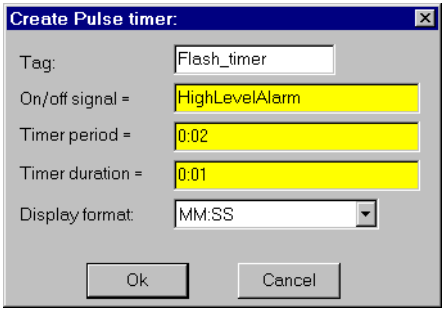


Figure 18-97. Pulse Definition Parameters Dialog Box

The **On/off signal** is a logical expression while **Timer period** and **Timer duration** are numeric expressions. Normally, these are simple time constants such as 0:20 (twenty seconds). See *Numeric Data Members* in Chapter 5, *Developer Tour*, for information on entering time constants.

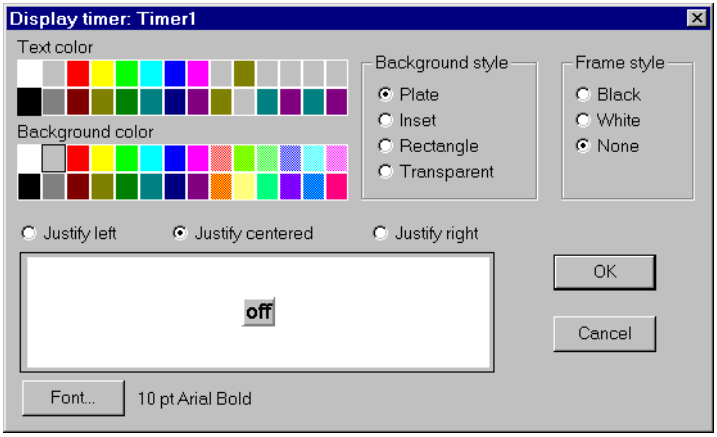


Figure 18-98. Pulse Display Parameters Dialog Box

Pulse Data Members

Table 18-92. Pulse Data Members

Data Members	Type	Read	Write	Description
(implicit)	logical	yes	no	Logical timer value

Comments ***Pulse can be used to periodically open a valve for a specified time duration. It can also act as a flasher to turn text and graphic signals on and off for display purposes.***

Related Objects ***DelayOff, DelayOn, Interval, OneShot, TimeOfxxx***

Pushbutton

Pushbutton generates a logical signal for receipt by other objects. A pushbutton changes state when you position the cursor over it and press the mouse button, trackball, touchscreen, or space bar. The pushbutton remains depressed and the output signal remains high until you release the button. If a **Verify on** message is defined, the operator must first acknowledge the message, then the output signal goes high, *but only momentarily*.

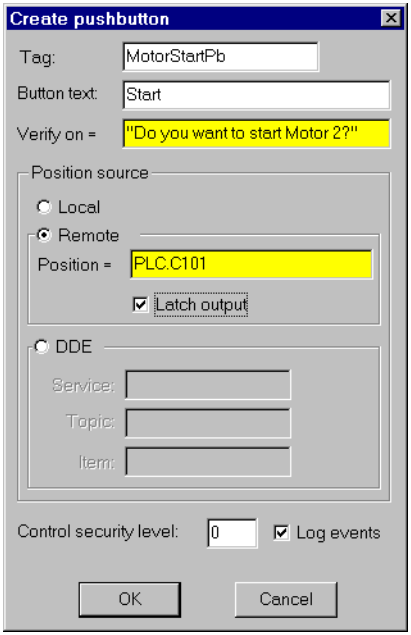


Figure 18-99. Pushbutton Definitions Parameters Dialog Box

Button text displays the specified text on the pushbutton.

Use **Verify on** to create a dynamic text expression to be displayed in a message dialog box. See Chapter 10, *Security*, for more information on security.

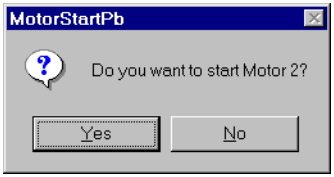


Figure 18-100. Verification Message Dialog Box

Position source determines where the value of the pushbutton resides. **Local** indicates the value of the pushbutton lies within the pushbutton itself—on the control panel. If the pushbutton is not depressed its signal is OFF, if depressed its signal is ON.

Remote pushbuttons get their values from a remote source, often the register in a controller they are connected to. Depressing the pushbutton changes the status of the register, and changing the status of the register depresses the pushbutton.

The **Remote** option is especially useful when you want to prevent Lookout from changing the value of setpoints or registers upon initial startup, or reconnection of lost communication. When you use this style of pushbutton you are creating a sort of looped signal. Half the loop is formed when you connect the controller register to the pushbutton with the Position expression, while the second half is formed when you connect the pushbutton output signal to the controller register. Position is a logical expression. Do not forget to complete the second half of the loop with the **Object»Edit Connections…** command.

When you select the **Remote** option, you can choose whether or not the pushbutton latches its output. The **Latch output** check box configures Lookout for controlling a latching-relay.

When a user clicks on a pushbutton that has latching selected, the pushbutton remains depressed, sending an ON signal (TRUE or high) until the Remote Position signal turns ON. Assume for example that an operator clicks on `MotorStartPb`, configured above. The pushbutton remains pushed in, sending a TRUE signal, until PLC.C101 goes TRUE. As soon as PLC.C101 goes TRUE, the pushbutton releases.

Much like Remote pushbuttons, **DDE** (Dynamic Data Exchange) pushbuttons get their values from a remote source. This could be a cell in a spreadsheet, another DDE aware application, or a second copy of Lookout running on the network. See Chapter 13, *Dynamic Data Exchange*, for information on **Service**, **Topic**, and **Item** parameters.

Control security level specifies the minimum security level operators must have to gain access to this individual object, and thus control it. See Chapter 10, *Security*, for more information on security.

The **Log events** option creates a permanent audit trail for the object—who did what and when. Any depression of the pushbutton is recorded to disk, including the time the button was depressed, and the operator's account name. See Chapter 11, *Logging Data and Events*, for more information on logging events.

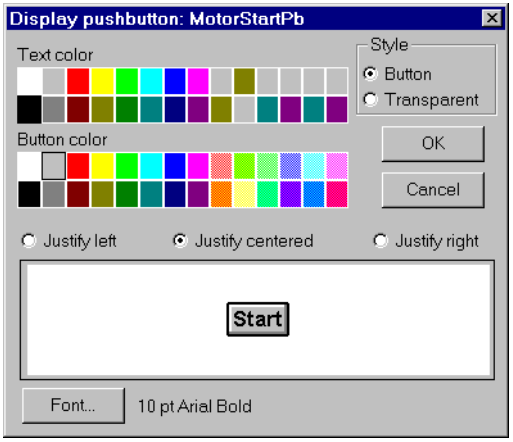


Figure 18-101. Pushbutton Display Parameters Dialog Box

Pushbutton Data Members

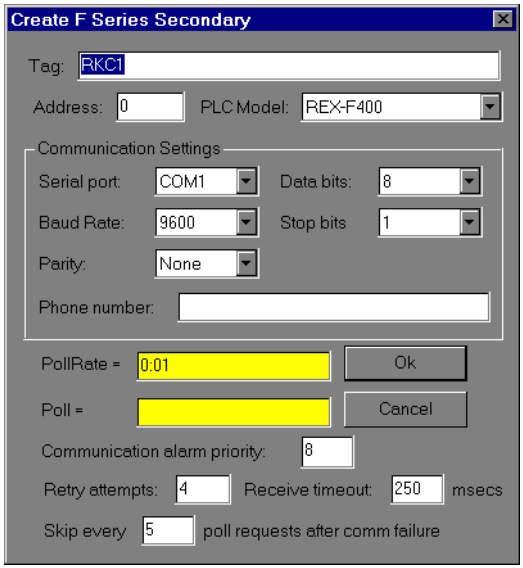
Table 18-93. Pushbutton Data Members

Data Members	Type	Read	Write	Description
(implicit)	logical	yes	no	Value of object (TRUE when button is depressed)
enable	logical	no	yes	If TRUE (the default), enables DDE. If FALSE, disables DDE. The default value is on. This input is ignored for non-DDE TextEntry objects.
visible	logical	no	yes	When FALSE, the pushbutton cannot be seen on the display panel. When TRUE, the button can be seen and controlled.

RKC F Series

RKC is a protocol driver class Lookout uses to communicate with RKC F Series devices using 7-bit ASCII serial communications.

With this driver you can read and write to all predefined data points allowed by a particular F Series model. When you create an RKC object, you have immediate access to all the object data members. See the data member tables for more information on data members for this object.



Address specifies which RKC F Series device you are communicating with. This number is between 0 and 31, and is set on the F Series device.

PLC Model specifies what model of RKC F Series device you are using. This driver supports the following models: REX-F400, REX-F700, and REX-F900.

Serial port specifies which COM port the object uses to communicate with the external device. This does not specify the communication type. Communication type is determined by the **Options»Serial Ports...** command.

Data rate indicates the baud rate Lookout uses to communicate with the hardware device. This setting should match the selection made on the physical device.

Data bits indicates the number of data bits Lookout uses to communicate with the hardware device. This setting should match the selection made on the physical device.

Stop bits indicates the number of stop bits Lookout uses to communicate with the hardware device. This setting should match the selection made on the physical device.

Parity indicates the parity that Lookout uses to communicate with the hardware device. This setting should match the selection made on the physical device.

Phone number specifies the number to be dialed if the selected serial port is configured for dial-up.

PollRate is a numeric expression that determines how often to poll the device. The object then polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *Introduction*, for information on entering time constants.

Poll is a logical expression. When this expression changes from FALSE to TRUE, Lookout polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Communication alarm priority determines the priority level of alarms generated by the RKC object. Such alarms are typically related to communications with the physical device.

Retry attempts specifies the consecutive number of times Lookout attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the RKC object generates an alarm and releases the COM port. See Chapter 6, *Serial Communications*, for more information.

Receive timeout is the time delay Lookout uses in waiting for a response from a device before retrying the request.

The **Skip every...** setting instructs Lookout to not poll a device it has lost communication with on every scheduled poll. Instead, Lookout skips the device in the polling cycle. Once communications have been reestablished, the device is polled on its regular cycle.

RKC Data Members

As with all Lookout drivers, you can access I/O points and other data through data members. The Lookout RKC object class currently supports the data members contained in the following tables, which are divided into groups according to their functionality.

Table 18-94. RKC Data Member Group

Parameter Group	Description
Measured Value Group	measured inputs and alarm status
Operation Mode Group	operation mode status
Memory Area and Set Value Group	set value (SV),
Parameter Group 10	measured input parameters
Parameter Group 11	remote setting input parameters
Parameter Group 12	output parameters
Parameter Group 13	auto-tuning bias parameters
Parameter Group 14	alarm 1 parameters
Parameter Group 15	analog output parameters
Parameter Group 16	positioning, proportioning, PID action parameters
Parameter Group 17	bar graph parameters
Parameter Group 20	input selection parameters
Parameter Group 21	setting parameters
Parameter Group 22	output action parameters
Parameter Group 23	alarm 2 parameters
Parameter Group 40	data lock parameters
Lookout data members	standard Lookout data members

☞ **Note** ***For a more complete definition of the function of these data members, see your RKC F Series documentation.***

Not all data members are valid for every F Series device. See your RKC documentation for which data members are valid for particular model numbers.

Table 18-95. Measured Value Group

Data Member	Type	Read	Write	Description
M1	numeric	yes	no	Measured value input
AA	logical	yes	no	First alarm status
AB	logical	yes	no	Second alarm status
AC	logical	yes	no	Heater break alarm status
O1	numeric	yes	no	Manipulated output
O2	numeric	yes	no	Cooling-side manipulated output
B1	logical	yes	no	Burnout status
B2	logical	yes	no	Burnout status of feedback resistance input
S2	numeric	yes	no	Remote set value
M2	numeric	yes	no	Feedback resistance input
M3	numeric	yes	no	Current transformer input
MS	numeric	yes	no	Set value

Table 18-96. Operation Mode Group

Data Member	Type	Read	Write	Description
J1	logical	yes	yes	Auto/manual mode
C1	logical	yes	yes	Local/remote mode
E1	logical	yes	yes	Local/external memory area transfer
ZA	numeric	yes	yes	Control area
G1	logical	yes	yes	PID control/autotuning
RA	logical	yes	no	Local/computer mode
SR	logical	yes	yes	RUN/STOP mode
ON	numeric	yes	no	Manipulated output

Table 18-97. Memory Area and Set Value Group

Data Member	Type	Read	Write	Description
S1:1 - S1:8	numeric	yes	yes	Set value
A1:1 - A1:8	numeric	yes	yes	First alarm setting
A2:1 - A2:8	numeric	yes	yes	Second alarm setting
P1:1 - P1:8	numeric	yes	yes	Proportional constant
I1:1 - I1:8	numeric	yes	yes	Integral constant
D1:1 - D1:8	numeric	yes	yes	Derivative constant
CA:1 - CA:8	numeric	yes	yes	Control response designation parameter
P2:1 - P2:8	numeric	yes	yes	Cooling-side proportional band
V1:1 - V1:8	numeric	yes	yes	Deadband
HH:1 - HH:8	numeric	yes	yes	Setting change rate limit

Table 18-98. Parameter Group 10 (Measured Input Parameters)

Data Member	Type	Read	Write	Description
PB	numeric	yes	yes	Measured value bias
F1	numeric	yes	yes	Measured value first order lag filter
DP	numeric	yes	yes	Low input cut-off

Table 18-99. Parameter Group 11(Remote Setting Input Parameters)

Data Member	Type	Read	Write	Description
RR	numeric	yes	yes	Remote set value ratio
RB	numeric	yes	yes	Remote set value bias
F2	numeric	yes	yes	Remote set value first order lag filter

Table 18-100. Parameter Group 12 (Output Parameters)

Data Member	Type	Read	Write	Description
OH	numeric	yes	yes	Manipulated output high limit
OL	numeric	yes	yes	Manipulated output low limit
OQ	numeric	yes	yes	Shortest cooling output on time
PH	numeric	yes	yes	Increase in output change rate limit
PL	numeric	yes	yes	Decrease in output change rate limit
IV	numeric	yes	yes	Upper ON/OFF differential gap
IW	numeric	yes	yes	Lower ON/OFF differential gap
OE	numeric	yes	yes	Manual output at abnormality

Table 18-101. Parameter Group 13 (Auto-Tuning Bias Parameters)

Data Member	Type	Read	Write	Description
GB	numeric	yes	yes	Set value bias when autotuning is performed

Table 18-102. Parameter Group 14 (Alarm 1 Parameters)

Data Member	Type	Read	Write	Description
HA	numeric	yes	yes	First alarm differential gap
TD	numeric	yes	yes	First alarm timer setting
A3	numeric	yes	yes	Heater break alarm value
HB	numeric	yes	yes	Second alarm differential gap
TG	numeric	yes	yes	Second alarm timer setting

Table 18-103. Parameter Group 15 (Analog Output Parameters)

Data Member	Type	Read	Write	Description
LA	numeric	yes	yes	Analog output type
HV	numeric	yes	yes	High limit of analog output range
HW	numeric	yes	yes	Low limit of analog output range

Table 18-104. Parameter Group 16 (Positioning, Proportioning, PID Action Parameters)

Data Member	Type	Read	Write	Description
V2	numeric	yes	yes	Neutral zone
VH	numeric	yes	yes	Open/close output differential gap
SY	numeric	yes	yes	Action selection at feedback resistance input break

Table 18-105. Parameter Group 17 (Bar Graph Parameter)

Data Member	Type	Read	Write	Description
DA	logical	yes	yes	Bar graph display selection

Table 18-106. Parameter Group 20 (Input Selection Parameters)

Data Member	Type	Read	Write	Description
XI	numeric	yes	yes	Measured value input type
AV	numeric	yes	yes	High limit of abnormality
AW	numeric	yes	yes	Low limit of abnormality
WH	logical	yes	yes	High limit of abnormality action selection
WL	logical	yes	yes	Low limit of abnormality action selection
XV	numeric	yes	yes	High limit of input programmable range
XW	numeric	yes	yes	Low limit of input programmable range

Table 18-106. Parameter Group 20 (Input Selection Parameters) (Continued)

Data Member	Type	Read	Write	Description
XU	numeric	yes	yes	Decimal point position selection
XH	logical	yes	yes	Square root extraction

Table 18-107. Parameter Group 21 (Setting Parameters)

Data Member	Type	Read	Write	Description
SH	numeric	yes	yes	High limit of setting range
SL	numeric	yes	yes	Low limit of setting range
XR	numeric	yes	yes	Remote setting input type
XL	logical	yes	yes	SV tracking selection

Table 18-108. Parameter Group 22 (Output Action Parameters)

Data Member	Type	Read	Write	Description
T0	numeric	yes	yes	Control output cycle setting
T1	numeric	yes	yes	Cooling-side output cycle setting
XE	logical	yes	yes	Direct/reverse action
XN	numeric	yes	yes	Action after power recovery setting
SX	numeric	yes	yes	Start determination point

Table 18-109. Parameter Group 23 (Alarm 2 Parameters)

Data Member	Type	Read	Write	Description
XA	numeric	yes	yes	First alarm action
NA	logical	yes	yes	First alarm energized/de-energized
OA	numeric	yes	yes	First alarm action when measured value exceeds abnormality
WA	numeric	yes	yes	First alarm hold action
XB	numeric	yes	yes	Second alarm action

Table 18-109. Parameter Group 23 (Alarm 2 Parameters) (Continued)

Data Member	Type	Read	Write	Description
NB	logical	yes	yes	Second alarm energized/de-energized
OB	numeric	yes	yes	Second alarm action when measured value exceeds abnormality
WB	numeric	yes	yes	Second alarm hold action

Table 18-110. Parameter Group 40 (Data Lock Parameters)

Data Member	Type	Read	Write	Description
LK	numeric	yes	yes	Data lock level setting
LL	logical	yes	yes	Memory area lock
DH	logical	yes	yes	Operation RUN/STOP display lock

Table 18-111. Lookout Data Members

Data Member	Type	Read	Write	Description
Update	logical	yes	no	Object-generated signal that pulses low each time it polls the device.
CommFail	logical	yes	no	Object-generated signal that is ON if Lookout cannot communicate with the device(s)
Poll	logical	no	yes	When this value transitions from FALSE to TRUE, Lookout polls the device.
PollRate	numeric	no	yes	Lookout expression that determines the device polling frequency.

RKC Status Messages

No response within timeout period

Lookout received no response from a device within the **Receive timeout** period. The RKC object is able to use the COM port, but when it polls the device, it does not respond—as if it is not even there. If you have daisy-chained several devices, you have introduced an inherent delay. You may have to significantly increase **Receive timeout** (and **Poll Rate**) to

ensure Lookout is allowing enough time to receive the expected response. This increase has nothing to do with the processing capabilities of Lookout. Rather it is based solely on **Data rate** and the number of devices on the chain. Also, verify your baud rate settings, cable connections, power, configuration settings, COM port settings, and polling addresses.

Invalid BCC in response
Lookout has received a frame with an invalid block check character (BCC). Check the cabling or look for two or more devices with the same address.

Invalid identifier in request
The request frame sent had an invalid identifier. This could possibly mean that you have requested an identifier that is not valid for your particular model in the RKC F Series. Check your model number carefully and read your RKC documentation to determine which identifiers are valid for which models.

Garbled or invalid frame
Lookout has received a frame without format characters in their proper positions. Check the **Receive gap** setting.

Numeric conversion failed
Lookout was unable to successfully convert ASCII data sent back from RKC into a number. Check the **Receive gap** setting.

No acknowledgment for write frame
There was no response to the write frame just sent. This could possibly mean that you have requested an identifier that is not valid for your particular model in the RKC F Series. Check your model number carefully and read RKC documentation to determine which identifiers are valid for which models.

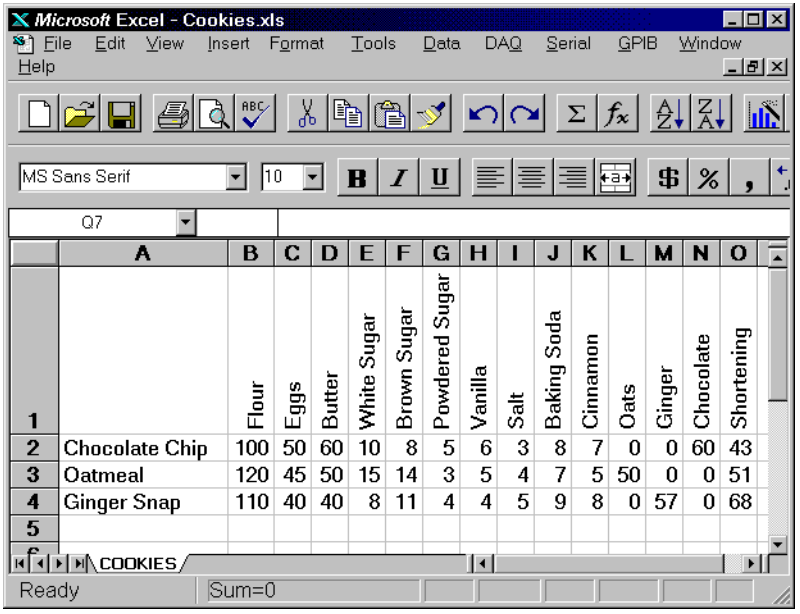
RKC F Series models supported:
REX-F400, REX-F700, REX-F900

Recipe

Recipe objects are an efficient means of importing large arrays of data (namely recipes and their ingredients) into Lookout using an Excel (`.xls`) spreadsheet. Once created and implemented, the operator can easily and quickly change the current recipe with the click of the mouse, thus selecting a new set of ingredients.

The best way to describe how the Recipe class works is to step through a typical example, in this case involving cookie manufacture.

There are two steps to creating and implementing a recipe object. First, you define your recipes with their respective ingredients in a spreadsheet program such as Excel (anything that creates an `.XLS` file will work—including Lotus 123). You can define up to 1,000 recipes in a single `.XLS` file. Each recipe can have up to 255 ingredients. Three cookie recipes are defined in this spreadsheet.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1		Flour	Eggs	Butter	White Sugar	Brown Sugar	Powdered Sugar	Vanilla	Salt	Baking Soda	Cinnamon	Oats	Ginger	Chocolate	Shortening
2	Chocolate Chip	100	50	60	10	8	5	6	3	8	7	0	0	60	43
3	Oatmeal	120	45	50	15	14	3	5	4	7	5	50	0	0	51
4	Ginger Snap	110	40	40	8	11	4	4	5	9	8	0	57	0	68
5															

The first row of the spreadsheet is reserved for ingredient names. They begin in column B. These ingredient names later become alias data members of the Recipe object. Therefore, ingredient names must be unique. They cannot have the same name as a native data member (see the Recipe Data Member table for data member names). Also, you cannot

name an ingredient `missing`; this is a reserved word. Valid characters in an ingredient name include A – Z, 0 – 9, the dollar sign ($), and a period (.). If you enter `Hi_@#!!There` as an ingredient name, Lookout names the alias `HiThere`. Alias names are case sensitive.

Beginning in Row 2, Column A lists the names of the various recipes. Recipes follow the same naming convention as ingredients. Each recipe is followed by its unique ingredient values.

Ingredient values can represent process inputs, parameters, and outputs. Process inputs typically represent raw materials and other inputs consumed in the batch process. Examples include the number of eggs consumed, amount of flour used, amount of energy consumed, possibly even work hours required or amount of traceable fixed costs consumed.

Another type of ingredient value is a process parameter. Process parameters identify operational settings such as furnace cooling time, an air pressure setpoint, or a Low pH alarm limit. Process parameters might also include identifications of specific equipment to be used during the batch process.

The third type of ingredient value is a process output. Such an ingredient value might represent the number of finished cookies expected from the batch, amount of byproduct expected, or a cost variance calculation based on the selected recipe.

Ingredient value quantities may be specified as constants or as equations based on other formula parameters such as batch size.

The second part of defining a recipe involves defining a recipe object in Lookout.

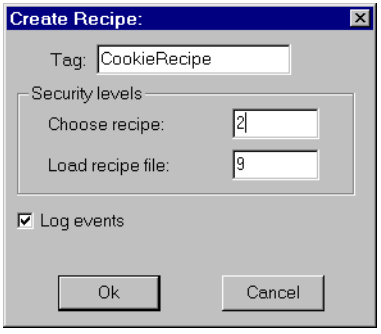


Figure 18-102. Recipe Definition Parameters Dialog Box

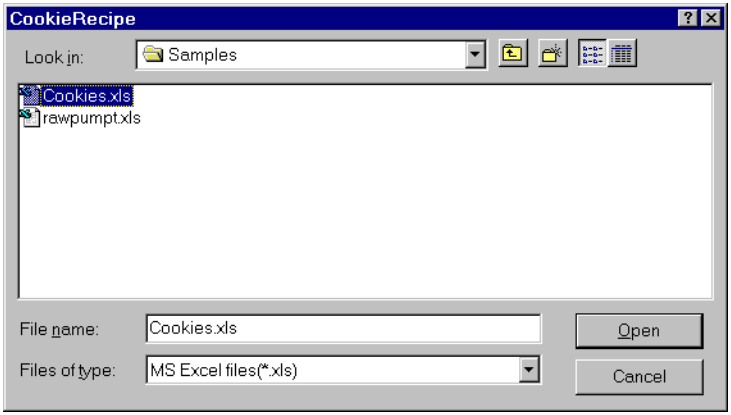
The first recipe dialog box defines security and event data logging.

Choose recipe security level specifies the minimum security level an operator must have to be able to select a recipe from all the recipes listed in the currently selected spreadsheet file.

Load recipe file security level specifies the minimum security level an operator must have to be able to select a different spreadsheet file.

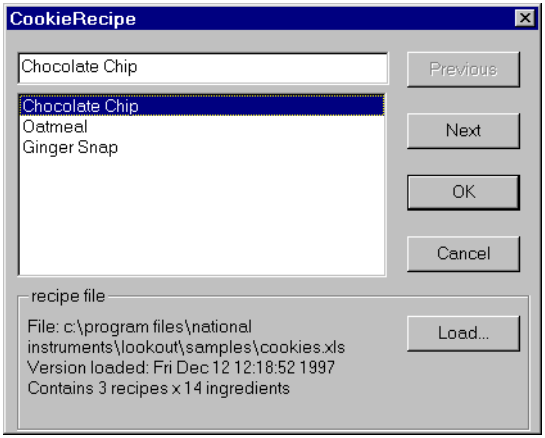
The **Log events** option creates a permanent audit trail for the object—who did what and when. Any selection of a different recipe or recipe file is logged to disk, including the time the action occurred and the operator's account name. See Chapter 11, *Logging Data and Events*, for more information on logging events.

After defining security and event data logging, Lookout presents you with a file selection dialog box.



Select the `cookies.xls` spreadsheet file because it has the three batch recipes in it.

Once a file is selected, Lookout presents a list of recipes.



As you can see, the recipe names come from column A of the spreadsheet. You can use the ***Previous*** and **Next** buttons to identify a recipe from among the list, or type the name in the data entry field above the list. (This same dialog box appears later when an operator clicks on the pushbutton representing this object.) Click on **OK** to choose the recipe you want to use. Select the recipe for oatmeal cookies.

The recipe file **Load** button invokes the file list dialog box, described previously. Click on this button to select a new `.xls` file.

☞ **Note** ***If the recipe is changed in the spreadsheet, the change is noted in the recipe file dialog box—but the values currently resident in the object data members remain intact. The operator must load the spreadsheet again to update the copy of the recipe file in Lookout. If you select a new*** `.xls` ***file to load but click on the Cancel button, it does not update! You must select*** **OK** ***for the recipe to actually be loaded.***

After loading the spreadsheet, Lookout presents the display parameters box.

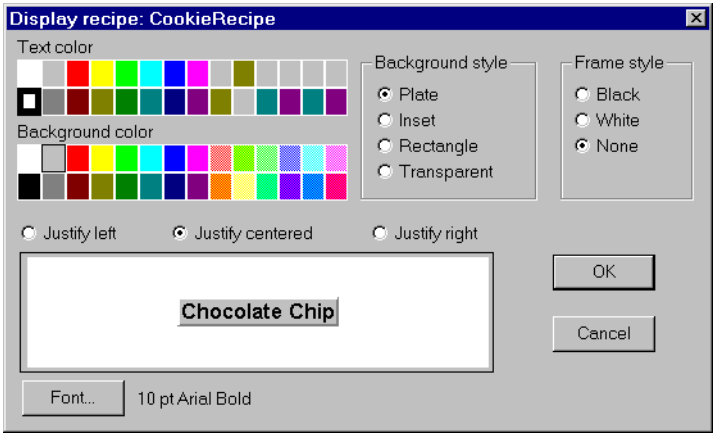
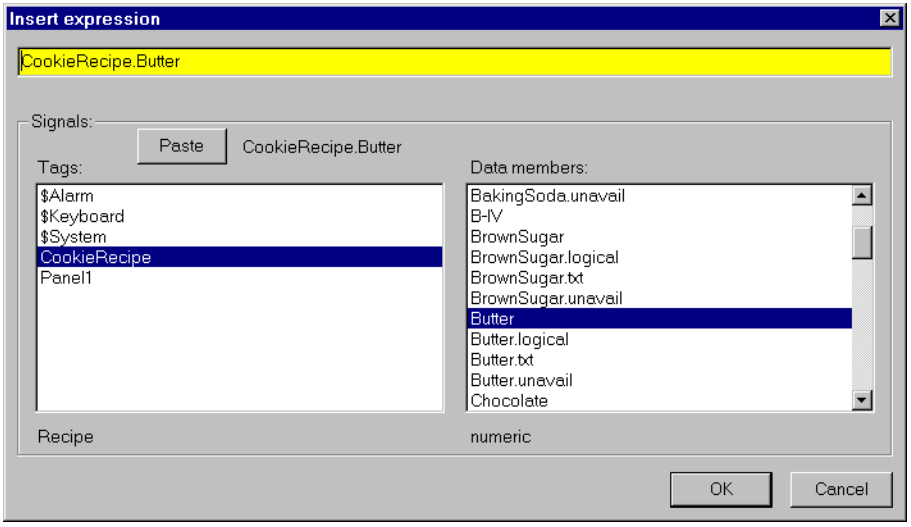


Figure 18-103. Recipe Object Display Parameters Box

After you choose the object display parameters, you can paste it into the panel.

When you select a new recipe, Lookout writes the ingredient values for the selected recipe into the corresponding data members of the object.

The actual number of data members that a recipe object has is based on the number of ingredients within it. This is best demonstrated by looking at the Insert Expression dialog box.



Notice that there are four data members for each defined ingredient. Actual data member names vary from object to object, depending upon your recipe ingredients. However, the four readable data member types for each ingredient are consistent.

Recipe Data Members

Table 18-112. Recipe Data Members

Data Member	Type	Read	Write	Description
(implicit)	text	yes	no	Name of currently selected recipe
B – IV	numeric	yes	no	Each letter, B through IV, represents a column in the spreadsheet. The value of the data member is the numeric amount of the ingredient for the currently selected recipe.
B.logical through IV.logical	logical	yes	no	Each letter, B through IV represents a column in the spreadsheet. Returns TRUE (ON) if the amount of the ingredient in the spreadsheet cell for the selected recipe is greater than zero. Returns FALSE if the specified amount for the ingredient is zero.
B.txt – IV.txt	text	yes	no	Each letter, B through IV represents a column in the spreadsheet. The value of the data member is the textual amount of the ingredient for the currently selected recipe.
B.unavail through IV.unavail	logical	yes	no	Each letter, B through IV represents a column in the spreadsheet. Returns TRUE if the spreadsheet cell is empty. Returns FALSE if the cell contains data.
pick1 through pick1000	logical	no	yes	Upon transition from FALSE to TRUE, chooses the respective recipe within the spreadsheet. When used with pushbutton objects, these data members can eliminate the need for operators to use the recipe list dialog box.

Comments ***The recipe object reads a block of continuous columns. Therefore, the ingredient names should be a contiguous list in Row 1. If a recipe does not use a particular ingredient, just leave the respective cell blank.***

When Lookout encounters a blank cell in Column A, it ignores the entire row. Thus, you can easily annotate your recipes by leaving a cell in Column A blank and adding text to the cell in Column B of the same row.

Reliance

Reliance is a protocol driver class Lookout uses for communicating with Reliance AutoMate controllers.

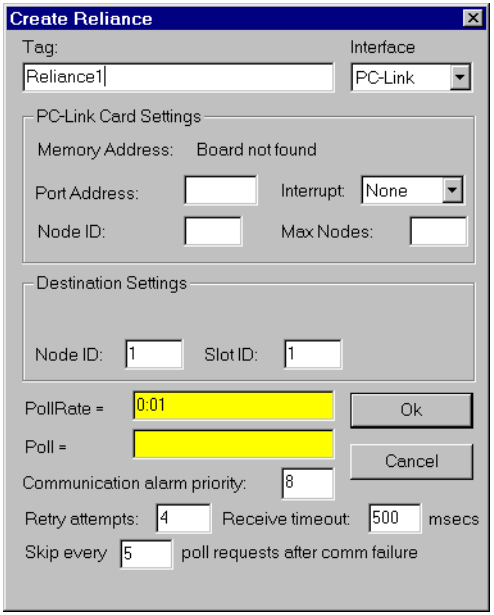


Figure 18-104. Reliance Definition Parameters Dialog Box

Note ***If you are developing your application without a PC-Link card installed in your computer, and do not know the settings in the target system, you can enter 250 for Port Address, 0 for Node ID, and 3 for Max Nodes. You can then create the Reliance object and correct the parameter values for the actual system later, if necessary.***

Interface is the method the object uses to communicate with the Reliance PLCs. Currently, only the Reliance R-Net is supported, using the Reliance PC-Link card.

PollRate is a numeric expression that determines how often to poll the device. Reliance then polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *Introduction*, for information on entering time constants.

Poll is a logical expression. When this expression changes from FALSE to TRUE, Lookout polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Communication alarm priority determines the priority level of object-generated alarms (0 – 10).

Retry attempts specifies the consecutive number of times Lookout attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the object generates an alarm. See Chapter 6, *Serial Communications*, for more information.

Receive timeout is the time delay Lookout uses in waiting for a response from a device before retrying the request.

The **Skip every…** setting instructs Lookout not to poll a device it has lost communication with on every scheduled poll. Instead, Lookout skips the device in the polling cycle accordingly. Once communications have been reestablished, the device is polled on its regular cycle.

PC-Link Card Settings

Port Address is the I/O port address used to access the software module on the card. See your PC-Link card documentation for specific settings.

Interrupt identifies the interrupt (IRQ) setting of your PC-Link card. The card generates an interrupt recognized by Lookout any time it receives an response from the device.

Node ID is the node ID setting of the PC-Link card on the R-Net.

Max Nodes is the maximum number of nodes on the R-Net.

Destination Settings

Node ID is the node ID on the R-Net setting of the AutoMate processor that controls the target AutoMate PLC.

Slot ID is the ID of the PLC rack slot in which the target AutoMate resides.

Reliance Data Members

All readable and writable members (inputs/outputs), polling instructions, read/write blocking, serial port usage, and so on, are bundled with the object. Therefore, as soon as you create an Reliance object you

immediately have access to all the object data members (see data member list).

☞ **Note** ***Lookout protocol driver objects automatically generate an efficient read/write blocking scheme based on the inputs and outputs being used in your process file. You are not required to build your own I/O blocking table.***

Reliance Data Members

Table 18-113. Reliance Data Members (all addresses are in octal)

Data Member	Type	Read	Write	Description
0 – 157775	numeric	yes	yes	16-bit input registers encoded as unsigned binary integers ranging from 0 to 65,535.
0.0 – 15775.17	logical	yes	yes	Access individual bits in holding registers and read them as logical ON/OFF values. The least significant bit is 1; the most significant bit is 17.
CommFail	logical	yes	no	Driver-generated signal that is ON if Lookout cannot communicate with the device.
D0 – D157774	numeric	yes	yes	Wide data format ranges from –99,999,999 to +99,999,999.
Poll	logical	no	yes	When this expression transitions from FALSE to TRUE, Lookout polls the device.
PollRate	numeric	no	yes	Lookout expression that determines the polling frequency of the device.
Update	logical	yes	no	Driver-generated signal that pulses each time the driver polls the device.

Reliance Status Messages

See your Reliance or PC-Link documentation for details of Reliance device generated messages.

Run

You can use Run objects to start an external program file from within Lookout. When the result of the **Run when** logical expression goes TRUE, the object executes the **Command line**.

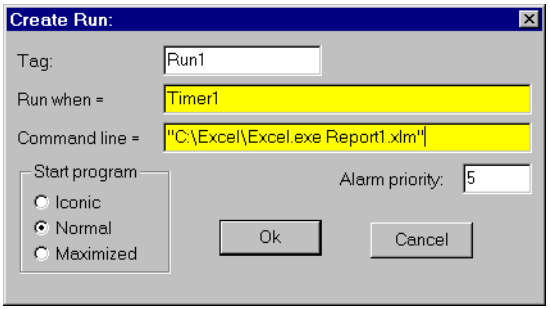


Figure 18-105. Run Definition Parameters Dialog Box

In this example, Lookout runs an Excel macro called `REPORT1.XLM` when the logical value returned by `Timer1` goes TRUE. `Timer1` is a TimeofDay object that triggers the report to run every day at 8:00 a.m.

The **Command line** text expression must be enclosed in quotation marks as shown. Notice that the example includes the full path name of the executable file. Ensure that your command line meets DOS syntax requirements. Because this is an expression data field, the command could be the result of a text expression.

You can specify how an application presents itself when you activate it with the **Start program** selections. If you select **Normal**, the application window appears in front of the open Lookout window when it is activated. If you want to reduce the application to an icon each time you start it, select **Iconic**. If you select **Maximized**, the application window replaces the Lookout window on the screen. (Lookout is still running; you just cannot see it. Press <Alt+Tab> to switch between applications.)

Table 18-114. Run Data Members

Data Member	Type	Read	Write	Description
none	—	—	—	Run does not have any data members

Comments ***If the application does not automatically shut down, multiple instances of the program may be running because of previous starts. Over time, this can snowball to the point where Windows performance is severely hampered.***

If you want to execute DOS commands from within Lookout, put the commands in a DOS batch file (`.BAT`) and then identify the batch file in the Command Line.

S5_3964

S5_3964 is a protocol driver class Lookout uses to communicate with Siemens Simatic S5 PLCs using the 3964 or 3964R protocols.

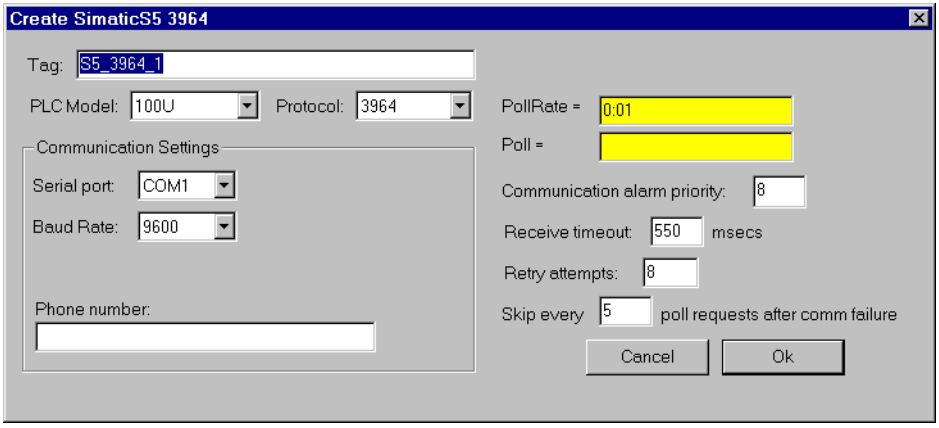


Figure 18-106. S5_3964 Definition Parameters Dialog Box

PLC Model specifies the model of Simatic S5 CPU the object communicates with.

Protocol specifies the protocol used to communicate with the PLC, 3964 or 3964R.

Serial port specifies which port the object uses for communication to the PLC.

Baud Rate specifies the speed at which the object communicates with the PLC.

Phone number specifies the number to be dialed if the serial port setting is configured for dial-up. This number only applies to the individual protocol object.

PollRate is a numeric expression that determines how often to poll the device. Normally, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *Introduction*, for information on entering time constants.

Poll is a logical expression. When this expression changes from FALSE to TRUE, Lookout polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Communication alarm priority determines the priority level of object-generated alarms (0 – 10).

Receive timeout is the time the object waits for a response from a device before retrying a request.

Retry attempts specifies the consecutive number of times Lookout attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the object generates an alarm and releases the COM port. See Chapter 6, *Serial Communications,* for more information.

The **Skip every…** setting instructs the object that, in the event of a communications failure, to skip the next specified number of polls to the PLC before attempting to re-establish communications. Once communications have been re-established, the device is polled on its regular cycle.

S5_3964 Data Members

This protocol driver object contains a great deal of data. All readable and writable members (inputs/outputs), polling instructions, read/write blocking, serial port usage, and so on, are bundled with the object. As soon as you create a S5_3964 object, you immediately have access to all the data members of that object.

☞ **Note** ***Lookout protocol driver objects automatically generate an efficient read/write blocking scheme based on the inputs and outputs being used in your process file. You are not required to build your own I/O blocking table.***

The suffixes for the PLC data members below (KC, KF, and so on) follow the Siemens format for data suffixes. That is, KC for Counter format, KT for Timer format, KB for byte format, KF for fixed-point format, KG for floating-point format, and no suffix for those data members for which only one format is possible, such as bit fields, counters, and timers.

Table 18-115. S5_3964 Data Members

Data Member	Type	Read	Write	Description
CommFail	logical	yes	no	Object-generated signal that is ON if the object cannot communicate with the PLC.
Poll	logical	no	yes	When this expression transitions from FALSE to TRUE, Lookout polls the device.
PollRate	numeric	no	yes	Lookout expression that determines the device polling frequency.
Update	logical	yes	no	Object-generated signal that pulses low each time it polls the device
A0KC – A65535KC	numeric	yes	no	Absolute address.
A0KF – A65535KF	numeric	yes	no	Absolute address.
A0KF – A65535KT	numeric	yes	no	Absolute address.
C0 – C255	numeric	yes	no	Counter.
DB1D0.0 – DB255D255.15	logical	yes	no	A bit in a Data Block word.
DB1DL0KB – DB255DL255KB	numeric	yes	no	The left byte in a Data Block word.
DB1DR0KB – DB255DR255KB	numeric	yes	no	The right byte in a Data Block word.
DB1DW0KC – DB255DW255KC	numeric	yes	yes	Word in a Data Block.
DB1DW0KF – DB255DW255KF	numeric	yes	yes	Word in a Data Block.
DB1DW0KT – DB255DW255KT	numeric	yes	yes	Word in a Data Block.
DB1DD0KG – DB255DD254KG	numeric	yes	yes	A double-word in a Data Block.
DX1D0.0 – DX255D255.15	logical	yes	no	A bit in an Extended Data Block word.

Table 18-115. S5_3964 Data Members (Continued)

Data Member	Type	Read	Write	Description
DX1DL0KB – DX255DL255KB	numeric	yes	no	The left byte in an Extended Data Block word.
DX1DR0KB – DX255DR255KB	numeric	yes	no	The right byte in an Extended Data Block word.
DX1DW0KC – DX255DW255KC	numeric	yes	yes	Word in an Extended Data Block.
DX1DW0KF – DX255DW255KF	numeric	yes	yes	Word in an Extended Data Block.
DX1DW0KT – DX255DW255KT	numeric	yes	yes	Word in an Extended Data Block.
DX1DD0KG – DX255DD254KG	numeric	yes	yes	A double-word in an Extended Data Block.
F0.0 – F255.7	logical	yes	no	A bit in Flag byte.
FY0KB – FY255KB	numeric	yes	no	A Flag byte.
FW0KF – FW254KF	numeric	yes	no	A Flag word.
FD0KG – FD252KG	numeric	yes	no	A Flag double-word.
I0.0 – I127.7	logical	yes	no	A bit in a byte of the Input (PII) data area.
IB0KB – IB127KB	numeric	yes	no	A byte of the Input (PII) data area.
IW0KF – IW126KF	numeric	yes	no	Word in of the Input (PII) data area.
ID0KG – ID124KG	numeric	yes	no	A double-word of the Input (PII) data area.
Q0.0 – Q127.7	logical	yes	no	A bit in a byte of the Output (PIO) data area.
QB0KB – QB127KB	numeric	yes	no	A byte of the Output (PIO) data area.
QW0KF – QW126KF	numeric	yes	no	Word in of the Output (PIO) data area.
QD0KG – QD124KG	numeric	yes	no	A double-word of the Output (PIO) data area.
RS0KC – RS511KC	numeric	yes	no	Word in the System data area.

Table 18-115. S5_3964 Data Members (Continued)

Data Member	Type	Read	Write	Description
RS0KF – RS511KF	numeric	yes	no	Word in the System data area.
RS0KT – RS511KT	numeric	yes	no	Word in the System data area.
T0 – T255	numeric	yes	no	Timer.

S5_3964 Alarms

Object Alarms
The following alarms originate in the object, and are only generated after the object has retried the request the number of times specified by the **Retry attempts** parameter.

No response from PLC
Lookout did not receive a response from the device within the **Receive timeout** period. The driver object is able to use the COM port, but when it polls the device, the device does not respond. You may have to increase **Receive timeout** to ensure Lookout is allowing enough time to receive the expected response. Also, verify your baud rate settings, cable connections, power, configuration settings, COM port settings, and polling addresses.

Unexpected response from PLC
A response was received from the PLC, but not the response expected according to the protocol.

Bad frame
A response frame was received from the PLC, but the frame is not valid according to the protocol. This is usually caused by a truncated frame. You may need to increase the **Receive Gap** setting in the **Options»Serial Ports…** dialog.

Bad BCC
The BCC computed by the object for a received frame did not match the BCC in the frame.

PLC Alarms
The following alarms originate in the PLC, and are generated immediately by the object. There are no retry attempts.

Illegal DB/DX number
The Data Block (DB) or Extended Data Block (DX) number in a read/write request to a PLC was not valid for the PLC.

Synchronization error
The PLC and S5_3964 object are not synchronized within the protocol. This usually happens after the object is modified and a read/write request was interrupted. After generating the alarm, the S5_3964 object attempts to resynchronize the protocol in the PLC.

Sample

The Sample object samples and holds data. Any time the **Sample** expression transitions from OFF to ON and the **Enable** expression is TRUE, the Sample object samples and stores a **Data** expression. Sample maintains an array of up to 35 previous samples. If **Enable** is left blank it is assumed to be TRUE. **Data** is a numeric expression while **Sample** and **Enable** are logical.

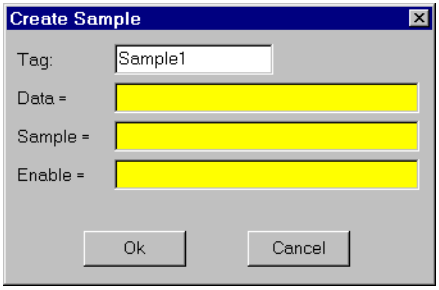


Figure 18-107. Sample Configuration Parameters Dialog Box

☞ **Note** ***Sample does not have a display parameters dialog box. You can display the result of a Sample output signal by using its data member in an expression.***

Sample Data members

Table 18-116. Sample Data Members

Data Member	Type	Read	Write	Description
(implicit)	numeric	yes	no	Current **Data** value—tracks **Data** input value.
1 – 35	numeric	yes	no	Previous samples. Signal 1 is the most recent sample since **Sample** went high.
DataReset	logical	no	yes	Upon transition from FALSE to TRUE, resets to zero all data members—including the current value and all previous samples.

Comments ***The*** *Reset* ***expression can be a regular pulse interval created by a TimeOfxxxx timer. For example, if you want to sample the temperature every hour of the day, use the output signal from a TimeOfHour timer in the*** *Reset* ***expression to sample the temperature at the beginning of each hour.***

Related Objects ***Minimum, Maximum, Average, SampleText***

SampleText

SampleText samples and stores the result of the **Data** expression any time the **Sample** expression transitions from OFF to ON and the **Enable** expression is TRUE. SampleText maintains an array of up to 35 previous samples. If **Enable** is left blank it is assumed to be TRUE.

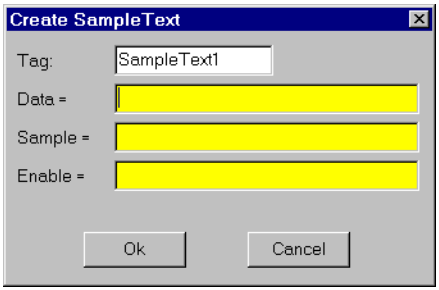


Figure 18-108. SampleText Definition Parameters Dialog Box.

Data is a text expression while **Sample** and **Enable** are logical expressions.

☞ **Note** ***SampleText does not have a display parameters dialog box. You can display the result of a Sample object output signal by using its data member in an expression.***

SampleText Data Members

Table 18-117. SampleText Data Members

Data Member	Type	Read	Write	Description
(implicit)	text	yes	no	Current **Data** value. Tracks **Data** input value.
1 – 35	text	yes	no	Previous samples. Signal 1 is the most recent sample since **Sample** went high.
DataReset	logical	no	yes	Upon transition from FALSE to TRUE, resets all data members—including the current value and all previous samples.

Related Objects ***Sample, TextEntry***

Scale

You can use the Scale object to create dynamic scales—that is, scales whose ranges and divisions can change based on numeric parameter expressions.

☞ **Note** ***If you want to create a simple scale that does not change dynamically (which is normally the case), use the Insert»Scale command described in Chapter 17,*** *Edit Mode Menu Commands****.***

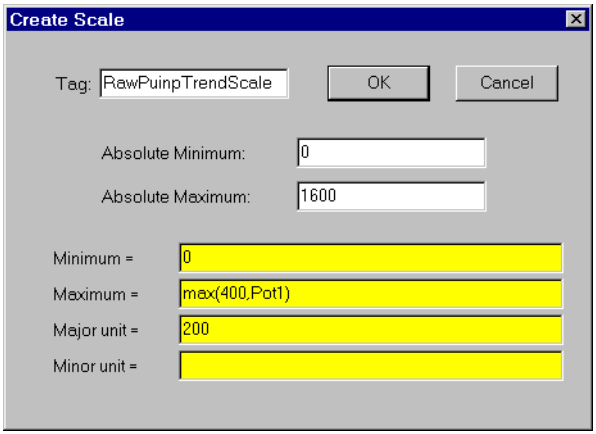


Figure 18-109. Scale Definition Parameters Dialog Box

Absolute Minimum and **Absolute Maximum** are numeric constants. They define the fullest possible range that the scale can show. These values act as clamps, restricting **Minimum** and **Maximum**.

Minimum and **Maximum** are numeric expressions you can use to change the minimum and maximum values on the scale. In the example above, the highest value of the scale (**Maximum**) is 400 if `Pot1` is less than 400, 1600 if `Pot1` is greater than 1600 (because of the **Absolute Maximum**), or equal to the value of Pot1.

Major unit specifies the number of units between major tick marks. **Minor unit** specifies the number of units between minor tick marks.

When you click on **OK**, the Display Scale dialog box appears.

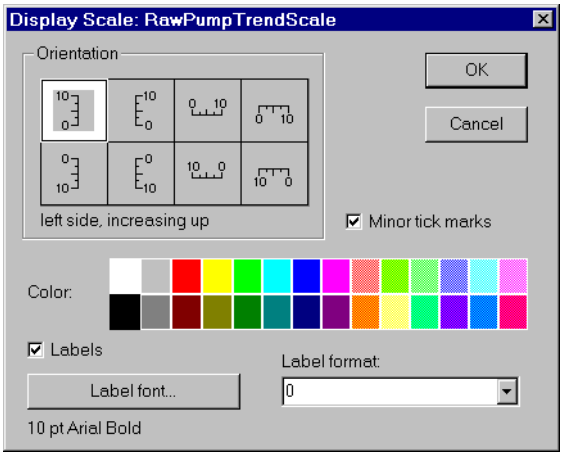


Figure 18-110. Scale Display Parameters Dialog Box

Specify **Orientation**, **Color**, **Label format**, and **Label font** as you choose.

You can remove minor tick marks by deselecting the **Minor tick marks** check box and you can remove label numbers from your scale altogether by deselecting the **Labels** check box. (Only major units have numeric labels.)

Scale Data Members

Table 18-118. Scale Data Members

Data Member	Type	Read	Write	Description
visible	logical	no	yes	When TRUE, the Scale becomes visible on the control panel. When FALSE, it is invisible. The default value is TRUE.

Comments ***Many people use this object class in conjunction with HyperTrends that are configured for a variable Y axis. They configure the*** *Minimum* ***and*** *Maximum* ***parameters of the Scale object to follow the same values as the*** *Max* ***and*** *Min* ***data members of the HyperTrend Object.***

SiemensTI505

SiemensTI505 is an Ethernet protocol driver object class Lookout uses to communicate with Siemens SIMATIC TI505 PLCs that are equipped with CP1431 NIMs.

☞ **Note** ***This object is available as a 16-bit object only, and only runs under Windows 3.1 and Windows 95.***

Designed using the SINEC TF software library, this object class uses the SINEC H1 protocol and fully conforms to the application layer of the Siemens SINEC H1-TF protocol stack.

Your PC must be equipped with a Siemens CP 1413 Ethernet communications card and Siemens TF-NET 1413 software. See *Configuring HI-TF* below for instructions on setting up your H1 driver.

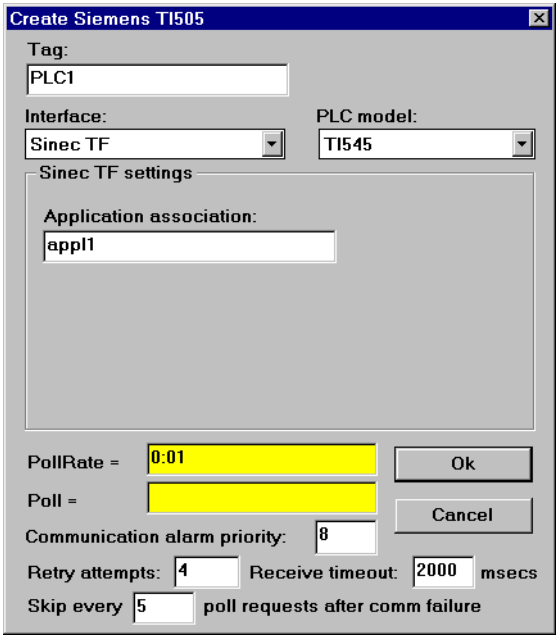


Figure 18-111. SiemensTI505 Definition Dialog Box

PLC Model specifies the PLC model number for the requested device. The list includes SIMATIC TI545, TI555 and TI565 PLCs.

Application association identifies an application name that you define using the `SINEC Com1413t.exe` program. This name identifies the

physical device that your object represents. See *Configuring HI-TF* below for more information.

PollRate is a numeric expression that determines how often to poll the device. SiemensTI505 then polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *Introduction*, for more information on entering time constants.

Poll is a logical expression. When this expression changes from FALSE to TRUE, Lookout polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Retry attempts specifies the consecutive number of times Lookout attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the object generates an alarm and releases the COM port. See Chapter 6, *Serial Communications*, for more information.

Receive timeout is the time delay Lookout uses in waiting for a response from a device before retrying the request.

Communication alarm priority determines the priority level of alarms generated by the SiemensTI505 object.

The **Skip every ___ polls** setting instructs Lookout not to poll a device it has lost communication with on every scheduled poll. Instead, Lookout skips the device in the polling cycle accordingly. Once communications have been reestablished, the device is polled on its regular cycle.

Configuring HI-TF

Install the Siemens TF-NET1413 MSDOS/Windows driver configuration software, including the COML 1413 TF configuration tool.

Configure the CP1413 Ethernet MAC address, memory map, and other settings by executing the following program from DOS (not Windows):

```
c:\sinec\bin\netinst.exe.
```

This creates a text file called `C:\SINEC\DATA\DOS_conf.dat` that stores the configuration parameters. You can later edit this file to change card settings in lieu of executing the install program.

If you are using a memory manager such as EMM386, add a memory exclude statement to your computer CONFIG.SYS file. For example, enter:

```
DEVICE=C:\WINDOWS\EMM386.EXE X=D000-DFFF
```

The CP 1413 Ethernet card uses 64 Kb of dual-ported RAM. Possible start addresses are D0000 and E0000.

☞ **Note** ***Be sure to verify that no other drivers are mapped to the selected memory location.***

From Windows, run C:\SINEC\COM\Com1413t.exe (the three-pawn icon) and configure the PLC name/address database that you download later to the CP 1413 card during system startup. Create a database file called STARTUP, and save it in the C:\SINEC\DATA directory.

This program creates two files, with your database represented by startup.Ldb and startup.txt. These files contain the application association names that you need for each PLC.

☞ **Note** ***There is a line of text in*** DOS_CONF.DAT ***that points to*** STARTUP.LDB***. To use a different path or filename for your*** .LDB ***file, modify*** DOS_CONF.DAT ***to point to the file you create.***

From Windows, activate C:\SINEC\H1\H1.exe (the single pawn icon) to create an H1 configuration file. Then, using the serial port on the NIM, download the configuration file to the H1 NIM in the PLC.

You must enter a unique Ethernet address and application association that matches one stored in the .LDB file. Also be sure that the Local TSAP and Remote TSAP values match exactly with the .LDB settings, except the Local TSAP and Remote TSAP values are swapped.

Create one H1 configuration file for each NIM in each PLC.

☞ **Note** ***Application Association, Local TSAP, Remote TSAP, and just about everything else in the Siemens software is case sensitive.***

Download the configuration to each NIM through the serial port on the NIM.

Edit your AUTOEXEC.BAT file and add the following command to the end of the file—but prior to any WIN command):

```
CALL C:\SINEC\BIN\STARTCP.BAT
```

Reboot your computer.

Within Lookout, create a SIEMENSTI505 object for each PLC using the appropriate Application Association. You are now ready to access PLC variables in Lookout.

SiemensTI505 Data Members

Protocol driver objects contain a great deal of data. All readable and writable members (inputs/outputs), polling instructions, read/write blocking, serial port usage, and so on, are bundled with the object. Therefore, as soon as you create a SiemensTI505 object you immediately have access to all the object data members (see data member list below).

☞ **Note** ***Lookout protocol driver objects automatically generate an efficient read/write blocking scheme based on the inputs and outputs being used in your process file. You are not required to build your own I/O blocking table.***

Table 18-119. SiemensTI505 Data Members

Data Member	Type	Read	Write	Description
AACK1 – AACK32000	numeric	yes	yes	(Analog Alarm) Alarm acknowledge flags
AADB1 – AADB32000	numeric	yes	yes	(Analog Alarm) Deadband
ACF1 – ACF32000	numeric	yes	yes	(Analog Alarm) Alarm C-flags
ACFH1 – ACFH32000	numeric	yes	yes	(Analog Alarm) Alarm C-flags most significant word
ACFL1 – ACFL32000	numeric	yes	yes	(Analog Alarm) Alarm C-flags least significant word
AERR1 – AERR32000	numeric	yes	yes	(Analog Alarm) Alarm error
AHA1 – AHA32000	numeric	yes	yes	(Analog Alarm) High alarm limit
AHHA1 – AHHA32000	numeric	yes	yes	(Analog Alarm) High high alarm limit
ALA1 – ALA32000	numeric	yes	yes	(Analog Alarm) Low alarm limit
ALLA1 – ALLA32000	numeric	yes	yes	(Analog Alarm) Low low alarm limit
AODA1 – AODA32000	numeric	yes	yes	(Analog Alarm) Orange deviation limit
APV1 – APV32000	numeric	yes	yes	(Analog Alarm) Process variable
APVH1 – APVH32000	numeric	yes	yes	(Analog Alarm) Process variable high limit
APVL1 – APVL32000	numeric	yes	yes	(Analog Alarm) Process variable low limit
ARCA1 – ARCA32000	numeric	yes	yes	(Analog Alarm) Rate of change limit
ASP1 – ASP32000	numeric	yes	yes	(Analog Alarm) Setpoint
ASPH1 – ASPH32000	numeric	yes	yes	(Analog Alarm) Setpoint high limit
ASPL1 – ASPL32000	numeric	yes	yes	(Analog Alarm) Setpoint low limit
ATS1 – ATS32000	numeric	yes	yes	(Analog Alarm) Sample rate
AVF1 – AVF32000	numeric	yes	yes	(Analog Alarm) Alarm flags

Table 18-119. SiemensTI505 Data Members (Continued)

Data Member	Type	Read	Write	Description
AYDA1 – AYDA32000	numeric	yes	yes	(Analog Alarm) Yellow deviation limit
C1 – C32000	logical	yes	yes	Control Registers
CommFail	logical	yes	no	Driver-generated signal that is ON if Lookout cannot communicate with the device for whatever reason
DCC1 – DCC32000	numeric	yes	no	Drum current count
DSC1 – DSC32000	numeric	yes	yes	Drum step current
DSP1 – DSP32000	numeric	yes	yes	Drum step preset
K1 – K32000	numeric	yes	yes	K-memory unsigned 16-bit integer value ranging from 0 to 65535
K1. – K32000.	numeric	yes	yes	K-memory 32-bit IEEE floating point value
K1D – K32000D	numeric	yes	yes	K-memory 32-bit unsigned integer value
K1S – K32000S	numeric	yes	yes	K-memory signed 16-bit integer value ranging from –32768 to 32767
LACK1 – LACK32000	numeric	yes	yes	(Loop) Alarm Acknowledge Flags
LADB1 – LADB32000	numeric	yes	yes	(Loop) Deadband
LCF1 – LCF32000	numeric	yes	yes	(Loop) C-flags
LCFH1 – LCFH32000	numeric	yes	yes	(Loop) C-flags most significant word
LCFL1 – LCFL32000	numeric	yes	yes	(Loop) C-flags least significant word
LERR1 – LERR32000	numeric	yes	no	(Loop) Error
LHA1 – LHA32000	numeric	yes	yes	(Loop) High alarm limit
LHHA1 – LHHA32000	numeric	yes	yes	(Loop) High high alarm limit
LKC1 – LKC32000	numeric	yes	yes	(Loop) Gain
LKD1 – LKD32000	numeric	yes	yes	(Loop) Derivative gain
LLA1 – LLA32000	numeric	yes	yes	(Loop) Low alarm limit

Table 18-119. SiemensTI505 Data Members (Continued)

Data Member	Type	Read	Write	Description
LLLA1 – LLLA32000	numeric	yes	yes	(Loop) Low low alarm limit
LM1 – LM32000	numeric	yes	yes	(Loop) Mode
LMN1 – LMN32000	numeric	yes	yes	(Loop) Output
LMX1 – LMX32000	numeric	yes	yes	(Loop) Bias
LODA1 – LODA32000	numeric	yes	yes	(Loop) Orange deviation limit
LPV1 – LPV32000	numeric	yes	yes	(Loop) Process variable
LPVH1 – LPVH32000	numeric	yes	yes	(Loop) Process variable high limit
LPVL1 – LPVL32000	numeric	yes	yes	(Loop) Process variable low limit
LRCA1 – LRCA32000	numeric	yes	yes	(Loop) Rate of change limit
LRSF1 – LRSF32000	numeric	yes	yes	(Loop) Ramp/Soak status flags
LS1 – LS32000	numeric	yes	no	(Loop) Status
LSP1 – LSP32000	numeric	yes	yes	(Loop) Setpoint
LSPH1 – LSPH32000	numeric	yes	yes	(Loop) Setpoint high limit
LSPL1 – LSPL32000	numeric	yes	yes	(Loop) Setpoint low limit
LTD1 – LTD32000	numeric	yes	yes	(Loop) Rate
LTI1 – LTI32000	numeric	yes	yes	(Loop) Reset
LTS1 – LTS32000	numeric	yes	yes	(Loop) Sample rate
LVF1 – LVF32000	numeric	yes	yes	(Loop) V-flags
LYDA1 – LYDA32000	numeric	yes	yes	(Loop) Yellow deviation limit
Poll	logical	no	yes	When this value transitions from FALSE to TRUE, the Lookout object polls the device
PollRate	numeric	no	yes	Specifies the frequency at which the Lookout object polls the device
TCC1 – TCC32000	numeric	yes	yes	(Analog Alarm) Timer/counter current
TCP1 – TCP32000	numeric	yes	yes	(Analog Alarm) Timer/counter preset

Table 18-119. SiemensTI505 Data Members (Continued)

Data Member	Type	Read	Write	Description
Update	logical	yes	no	Driver-generated signal that pulses each time the driver polls the device
V1 – V32000	numeric	yes	yes	V-memory unsigned 16-bit integer value ranging from 0 to 65535
V1. – V32000.	numeric	yes	yes	V-memory 32-bit IEEE floating point value
V1D – V32000D	numeric	yes	yes	V-memory 32-bit unsigned integer value
V1S – V32000S	numeric	yes	yes	V-memory signed 16-bit integer value ranging from –32768 to 32767
WX1 – WX32000	numeric	yes	no	Word Image Inputs—16-bit values that typically range from 6400 – 32000 for 4 – 20 mA signals, and 0 – 32000 for 0 – 5V signals.
WY1 – WY32000	numeric	yes	yes	Word Image Outputs—16-bit values that typically range from 6400 – 32000 for 4 – 20 mA signals, and 0 – 32000 for 0 – 5V signals.
X1 – X32000	logical	yes	no	Discrete Inputs—unassigned Xs may be used as control registers
Y1 – Y32000	logical	yes	yes	Discrete Outputs—same memory space as Discrete Inputs, so X37 references the same point as Y37. Unassigned Ys may be used as control registers

SiemensTI505 Status Messages

The following alarms are SINEC error codes returned by the Siemens software. For more detailed information, consult the Siemens SINEC TF documentation.

cannot initialize Sinec TF service

hardware error

invalid address

invalid app. assoc. name: *xxxx*

more than one object using same app. assoc.

no more space in PDU

no response

no response (reconnecting)

not available at times

object access not allowed

object attribute inconsistent

object does not exist

object not defined

object now invalid

type not supported

type/alt acc. not consistent

Sixnet

Sixnet is a protocol driver object class Lookout uses to communicate with Sixnet IOMUX RTUs (remote terminal units), Versamux RTUs, and DIN rail-mounted Sixtrak I/O modules.

The Lookout Sixnet object class establishes an interface to the Sixnet Control Room software using direct DLL calls (not DDE). Using this seamless connection between programs, Lookout can communicate with Sixnet devices through your serial port (RS-232 or RS-422/485), through Ethernet, or RTUnet.

☞ **Note** ***This protocol driver object class requires Version 1.0 or later of Sixnet Control Room I/O Map software, and Version 3.5 build 15 (or later) of Lookout.***

Create one Lookout Sixnet object for each station you define in the Control Room software. In order to make Sixtags names correspond as closely as possible to Lookout alias names, you should name the Lookout objects using the eight-letter prefixes of the Sixnet stations.

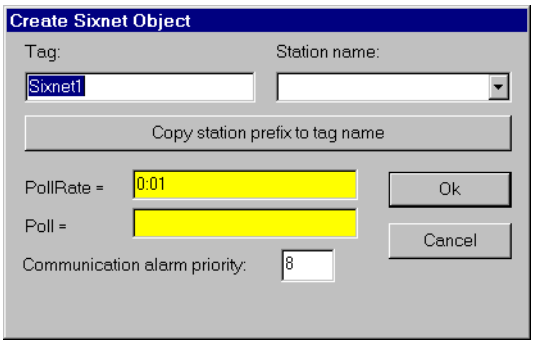


Figure 18-112. Sixnet Configuration Parameters Dialog Box

Station name is a pull-down list box of all stations defined in the Control Room software. Select the station that you want the object to represent.

PollRate is a numeric expression that specifies how often Lookout polls the Control Room software. Normally, this is a simple time constant such as 0:01 (one second), but you may choose to make this a complex expression, making the **PollRate** change dynamically based on criteria that you specify. See *Numeric Data Members* in Chapter 2, *Introduction*, for information on entering time constants and variables.

Poll is a logical expression. When this expression changes from FALSE to TRUE, Lookout polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Communication alarm priority determines the priority level of alarms generated by the Lookout Sixnet object. Such alarms are typically related to DLL handshaking with the Control Room software.

Sixnet Data Members

A Sixnet object represents all values within a given station; therefore, it can contain a great deal of data. The object can read and write all possible data point types, including predefined and user-defined I/O types. When you create a Sixnet object, you have immediate access to all data within the assigned station (see *data member* list below).

Each register in the Sixnet I/O Map corresponds to a data member within the Lookout Sixnet object. The register I/O type number and address are encoded into the name of the data member. Predefined I/O type numbers correspond with Lookout data member prefixes as follows:

I/O Type Number	Data Member Prefix
0	AI (Analog In)
1	AO (Analog Out)
10	DI (Discrete In)
11	DO (Discrete Out)
20	LongIn
21	LongOut
22	FloatIn
23	FloatOut

For user-defined I/O types, the Sixnet I/O type number is specified by the name of the Lookout data member which takes the following form:

<DataType><ioTypeNumber>:<Address>

where <DataType> is Bit, Byte, Short, Word, Long, Float, or Double; <ioTypeNumber> is a number between 0 and 126 inclusive; and <Address> is a number whose legal range depends on the data type. Thus the Lookout point `Sixnet1.Word33:99` corresponds to the hundredth register with I/O

type number 33 for the station with name `Sixnet1.Station`. The register is read as a word (that is, an unsigned 16-bit number).

You can use any data type with any I/O type number in Lookout. This means that you can read a long (32-bit signed number) from two consecutive analog registers. This makes it very easy to get the value of a 32-bit counter that is stored in consecutive 16-bit registers in a PLC. For example, the data member `Long0:7` would correspond to analog inputs `AI7` and `AI8` and it would be interpreted as a signed, 32-bit register. This capability also means that you have the choice of reading an analog value as either a signed value (using `AI0` or `Short0:0`) or as an unsigned value (using `Word0:0`).

☞ **Note** ***Writing a bit to an analog register sets that analog value to 0 or 1. It does not set just one of the bits to 0 or 1.***

Table 18-120. Sixnet Data Members

Data Member	Type	Read	Write	Description
AI0 – AI32499	numeric	yes	yes	Analog input encoded as a 16-bit signed integer ranging from –32768 to +32767
AO0 – AO32499	numeric	yes	yes	Analog output encoded as a 16-bit signed integer ranging from –32768 to +32767
Bit0:0 – Bit126:64999	logical	yes	yes	User-defined discrete I/O (TRUE or FALSE)
Byte0:0 – Byte126:64999	numeric	yes	yes	User-defined register encoded as an 8-bit unsigned integer ranging from zero to 255
CommFail	logical	yes	no	Driver-generated signal that is ON if Lookout cannot communicate with the device for whatever reason
DI0 – DI64999	logical	yes	yes	Discrete input (TRUE or FALSE)
DO0 – DO64999	logical	yes	yes	Discrete output (TRUE or FALSE)
Double0:0 – Double126:8124	numeric	yes	yes	User-defined I/O referring to 4 consecutive registers encoded as a single 64-bit floating point value

Table 18-120. Sixnet Data Members (Continued)

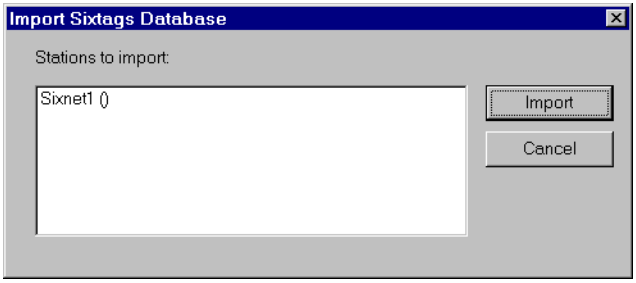
Data Member	Type	Read	Write	Description
Float0:0 – Float126:16299	numeric	yes	yes	User-defined register pair encoded as a 32-bit IEEE floating point value
FloatIn0 – FloatIn16299	numeric	yes	yes	32-bit IEEE floating point value —reads two adjacent registers as a single 32-bit floating point value
FloatOut0 – FloatOut16299	numeric	yes	yes	32-bit IEEE floating point value —writes two adjacent registers as a single 32-bit floating point value
Long0:0 – Long126:16299	numeric	yes	yes	User-defined register pair encoded as a signed 32-bit long integer. Lookout reads two adjacent registers as a single 32-bit number ranging from –2147483648 to +2147483647.
LongIn0 – LongIn16299	numeric	yes	yes	Long input encoded as a signed 32-bit long integer. Lookout reads two adjacent registers as a single 32-bit number ranging from –2147483648 to +2147483647.
LongOut0 – LongOut16299	numeric	yes	yes	Long output encoded as a signed 32-bit long integer. Lookout writes two adjacent registers as a single 32-bit number ranging from –2147483648 to +2147483647.
Poll	logical	no	yes	When this value transitions from FALSE to TRUE, Lookout polls the device.
PollRate	numeric	no	yes	Lookout expression that determines the device polling frequency.
Short0:0 – Short126:32499	numeric	yes	yes	User-defined register encoded as 16-bit signed integer ranging from –32767 to +32768
Station	text	yes	no	Station name (such as, Local Computer)

Table 18-120. Sixnet Data Members (Continued)

Data Member	Type	Read	Write	Description
Update	logical	yes	no	Driver-generated signal that pulses each time the driver polls the device
Word0:0 – Word126:32499	numeric	yes	no	User-defined register encoded as 16-bit unsigned integer ranging from 0 to 65535

Importing Sixtags Database

With the Sixnet Lookout object you can take advantage of Sixnet's Sixtags database. After you create at least one Sixnet object, the Sixnet class adds a menu selection (Import Sixtags database…) to the Lookout **Options** menu. Use this menu command to import a set of aliases for one or more stations.



When you select the menu command, a dialog box appears, listing all of the Lookout Sixnet objects—giving their tag names and corresponding station names. From the dialog box, select one or more station names and click on Import. Notice that spaces in the Sixtags names are replaced by underscores. You can re-import tag files as your Sixtags data is modified, and Lookout readjusts the aliased tag names automatically, in real time.

Sixnet Status Messages

Unable to load Sixnet IOMAP library: iodbase.dll
The Sixnet software isn't installed—Lookout cannot find the `iodbase.dll` library. Make sure that you have installed the Control Room software properly. This should put the `iodbase.dll` library into the `\WINDOWS\SYSTEM` directory.

No Sixnet configuration currently loaded
The Sixnet I/O Map software is not running yet. Use the Sixnet I/O Map program to open and run a project file. According to the Sixnet I/O Map help file, your DLL must be loaded and scanning before Lookout can control I/O. The easiest way to load the DLL and start scanning is to run the Control Room Power Switch. You could also select the **Run** command from the **Control** menu in the Sixnet I/O Map.

Station <name> is not on line
Sixnet reports that the named station is not on line. Lookout might still be able to read from and write to registers for that station, but the updates won't be propagated to the remote device.

Read error (<station name>, type <type number>, address <address>)
Write error (<station name>, type <type number>, address <address>)
One of the following reasons will be given:

- Address out of range
- Bad station, address, or type number
- Bad type number

Spinner

Spinner is a small, rotating disk. Its rotation speed can be variable, to represent the magnitude of a numeric **Signal**, or its rotation can be turned on or off based on the logical signal, **Spin**.

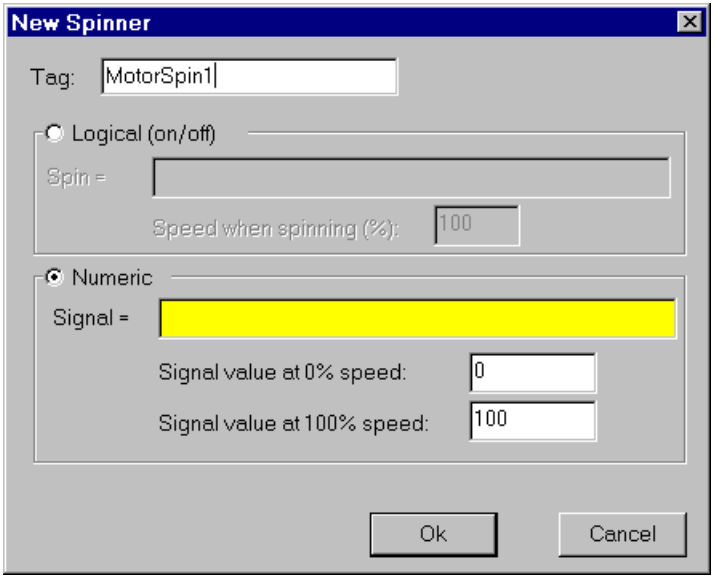


Figure 18-113. Spinner Definition Parameters Dialog Box

Logical (on/off) and **Numeric** choose whether the spinner responds to a logical signal or a numeric signal. Choose **Logical** if you want to be able to turn the spinner on and off. Choose **Numeric** if you want the speed and direction of the spinner to change depending on a numeric variable.

Spin is a logical expression. When the result of the logical expression is TRUE, the spinner rotates at the rate defined by the **Speed when spinning (%)** field. **Speed when spinning (%)** is a numeric constant, ranging from –100 to 100.

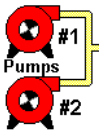
Connecting the spinner to a positive value rotates the spinner in a counterclockwise direction. A negative value rotates the spinner in a clockwise direction.

Signal is a numeric expression. The result of this expression dictates the spin speed based on the linear range defined by **Signal value at 0% speed** and **Signal value at 100% speed**.

Table 18-121. Spinner Data Members

Data Members	Type	Read	Write	Description
none	—	—	—	Spinner does not have any data members

Comments ***Spinners are typically used to represent flow through a line or to show a motor running.***



Spreadsheet

Spreadsheet permanently stores data to disk in spreadsheet files. You can log data on even and uneven intervals, when a data value changes, when an event occurs, or when any one of these things happen. Hence, you can implement complex logging criteria to meet almost any data storage need.

After each log, a new row is automatically added to the spreadsheet file. Lookout can log a new row of data approximately 10 times per second; however, the time stamps associated with each row are rounded to the nearest second.

Each spreadsheet file may store any number of data signals. Each data signal is assigned a spreadsheet column, beginning with column number two. The first column contains the date and time. The first row contains the expressions associated with the data in each column. You may create any number of Spreadsheet objects for a given process.

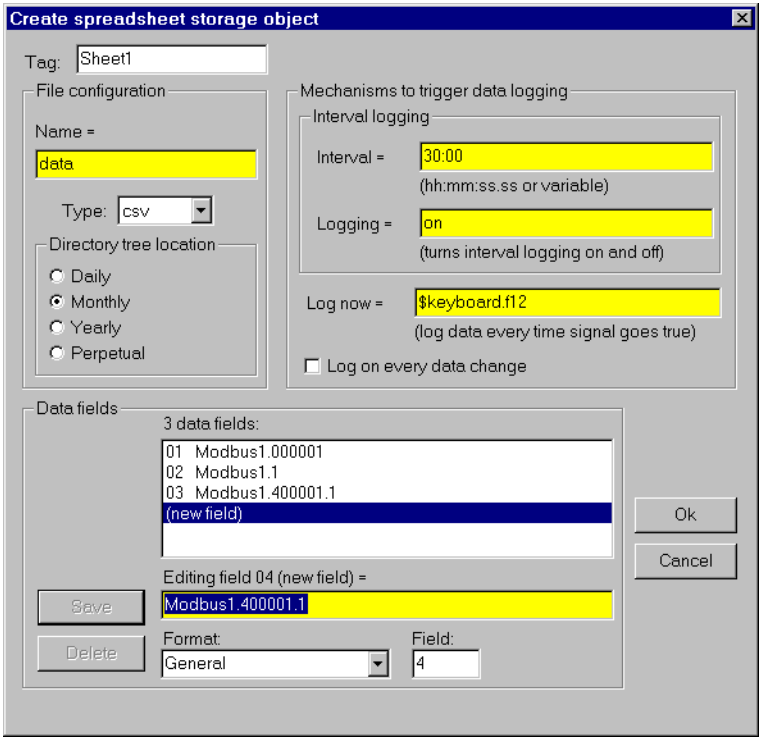


Figure 18-114. Spreadsheet Configuration Parameters Dialog Box

Name is the filename used to create a spreadsheet file. Lookout assigns a DOS filename to each spreadsheet file by adding the **Type** extension to the **Name**. Currently, Lookout supports only one **Type**: comma separated value format (`.csv`). Most database and spreadsheet programs including Microsoft Excel directly read the `.csv` file format.

Because the **Name** parameter is a text expression field, you can create new `.CSV` files with unique names dynamically. This is especially useful for recording batch processing data. The definition dialog box above is configured so that an operator can enter a batch name using a TextEntry object before the batch is started. The text expression appends the filename to the specified path, `C:\BATLOG\`. So if the operator enters a file name like `BATCH71`, then the full path name would be `C:\BATLOG\BATCH71.CSV`. When the `BatchRun` logical signal goes TRUE, Lookout creates the new `.CSV` file and begins writing to it. When `BatchRun` goes FALSE, logging ends, leaving a comprehensive log of all data associated with the batch.

The above example forces Lookout to store the `.CSV` file in a particular directory because it specifies a full path name. If you enter a relative pathname like `"\BATLOG\"&TextEntry1`, the file is located in that subdirectory of the identified **Directory tree location**. So, for example, the full path name of the file might be `C:\LOOKOUT\1995\SEP\BATLOG\BATCH71.CSV`.

If you enter just a filename such as `"DATA"`, the file location is based on the path specified by the **Directory tree location** selection. For example, the full path name of the file might be `C:\LOOKOUT\1995\SEP\DATA.CSV`.

The **Data files location** parameter in the System Options dialog box specifies the root directory under which all data logging should begin. Lookout stores alarm files in subdirectories under this root directory as specified by **Directory tree location**. See the **Options»System…** command in Chapter 16, *Runtime Menu Commands*, for more information.

If you select **Daily**, Lookout creates a new file and subdirectory every day in which to store the data. If you select **Monthly**, Lookout creates a new file and subdirectory every month in which to store the data. If you select **Yearly**, Lookout creates a new file and subdirectory every year. **Perpetual** files are stored in the root directory as specified by your **Data files location** parameter.

The following examples are the DOS filenames and directory trees created by Lookout for a spreadsheet file named, `DATA`. These examples assume your **Data files location** is set to `C:\LOOKOUT`.

Daily	Yearly
c:\lookout\1993\sep\09\data.csv \sep\10\data.csv \sep\11\data.csv	c:\lookout\1993\data.csv \1994\data.csv \1995\data.csv
Monthly	**Perpetual**
c:\lookout\1993\sep\data.csv \oct\data.csv \nov\data.csv	c:\lookout\data.csv

The **Mechanisms to trigger data logging** are a set of tools used to create a simple or complex logging scheme, as desired. Use these parameters to log data based on a timer, event, or any combination of the two. When the spreadsheet is triggered, all data in the Data fields is logged to disk.

Interval is a numeric expression used to create a Pulse timer with a pulse period of the specified time period and a pulse duration of zero. Normally this is a time formatted constant value such as 15:00 (fifteen minutes), for example.

Logging is a logical expression that turns the **Interval** parameter on and off. It could be a switch on a control panel, a logical input from an external device, or a more complex expression. Normally this is a constant value, `ON` or `OFF`.

Log now is a logical expression. When **Log now** transitions from OFF to ON, Lookout logs the data. A transition from ON to OFF has no effect. This expression could be a pushbutton on a control panel, a logical signal from a device, or a more complex expression.

The **Log on every data change** option should be used with care. When turned on, it triggers the logging of data any time any one of the data fields experiences a change. This is normally used to log the starting and stopping of pumps, opening and closing of valves, or other similar events. If your data fields contain even a single analog value that changes often, you could end up triggering the logger thousands of times. Or if they contain a logical value that changes frequently, you could have the same problem.

The **Data fields** window lists all expressions that have been entered for logging in the order of their field number.

The **Save** button saves your new or modified expression in the **Data fields** window along with a new field number if any. Normally, **Data fields** contain simple expressions like `PLC1.Tanklevel`.

The **Delete** button deletes the currently selected expression from the data fields list.

The **Format** option specifies the numeric format assigned to the currently selected numeric expression when it is logged to disk. This has no effect on logical or text expressions.

Field indicates the number of the currently selected data field.

☞ **Note** ***Field numbers should not be modified after data has been stored or the data will not appear under correct headers until a new file is created.***

Spreadsheet Data Members

Table 18-122. Spreadsheet Data Members

Data Members	Type	Read	Write	Description
logged	logical	yes	no	Spreadsheet file update pulse. The Spreadsheet object generates this logical pulse with a pulse duration of zero after each successful log.

SqlExec

SqlExec is an object Lookout uses to communicate with the ODBC driver. ODBC allows Lookout to connect to any database that supports ODBC.

SqlExec connects to ODBC using the standard ODBC calls, and queries the database using SQL statements. The data, if any, is returned in the object data members.

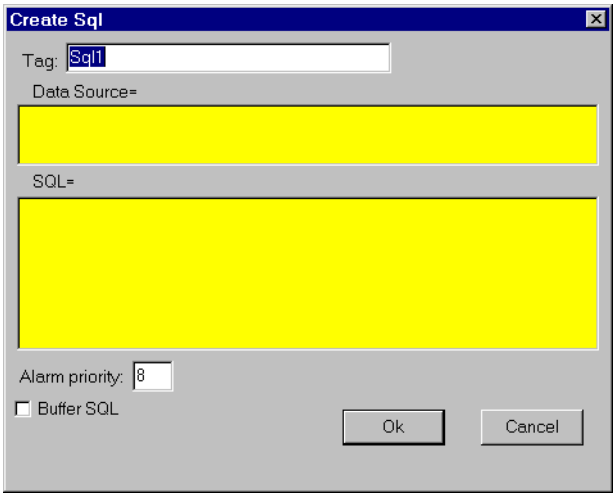


Figure 18-115. SqlExec Configuration Parameters Dialog Box

Data Source specifies the Data Source Name (DSN) as well as any other parameters needed by the ODBC driver to make the connection to the data source. For example if you want to connect to Excel 4.0, use

```
"DSN=Excel Files; DBQ=C:\pathname;"
```

SQL is the SQL string you want to pass to the ODBC driver.

Alarm Priority determines the priority level of object-generated alarms.

Buffer SQL tells the SqlExec object to buffer any entries that fail for that object. An entry consists of the Data Source string and the SQL string.

SqlExec Data Members

Table 18-123. SqlExec Data Members

Data Member	Type	Read	Write	Description
Execute	logical	no	yes	Executes the submitted SQL statement when it receives a change from low to high.
ReadOnly	logical	yes	yes	If TRUE, opens the file in read only mode. If FALSE, opens the file in read write mode.
Status	logical	yes	no	High when the object is processing a query.
Failure	logical	yes	no	High if the last submitted query failed.
c1 – c65535	numeric	yes	no	Value of the *n*th column in the row returned by ODBC.
c1.txt – c65535.txt	text	yes	no	Value of the *n*th column in the row returned by ODBC.
c1.logical – c65535.logical	logical	yes	no	Value of the *n*th column in the row returned by ODBC.

SqlExec Comments

The placement of the data in the data members is determined by the SQL string. If you submit the SQL string

```
"select Album, Artist, NumTracks from cdTable
where NumTracks>5"
```

the return value for Album is contained in the data member `c1.txt`, Artist in `c2.txt`, and NumTracks in `c3`. This value is displayed according to the data member you choose—Text, Logical, or Numeric. If you connect both to `c1` and `c1.txt`, you are connecting to the same value twice. In these circumstances, the value is represented in a different form.

A select statement only returns the first row that it encounters with the matching criteria. An update statement updates all the rows it encounters with the matching criteria.

If you connect your SqlExec object to a timer that pulses faster than several times per second, Lookout may become so busy handling SQL queries that it becomes unresponsive to user input from the mouse or keyboard.

Comma Separated Value (CSV) files are supported in ODBC. You can append data to them using the SQL insert command, but you cannot update records using the SQL update command. This is a limitation of the CSV file format.

- Excel files are accessible through ODBC, but they do have some unusual properties.
- If you use SqlExec to update an Excel 5.0 or greater workbook, the workbook must be closed and have multi-user editing turned off.
- If you are just trying to read from a workbook using a select query, then multi-user editing can be on, and the file can be opened or closed.
- If you try to access an Excel 5.0 or greater workbook while it is open and multi-user editing is off, Lookout locks up until the Excel file is closed. This is caused by a problem in the Excel ODBC driver.
- If you are using Excel 4.0 you can update or read from the worksheet only when it is closed. If you try to access an Excel 4.0 worksheet while it is open, you will receive an alarm in Lookout.

Citadel is also accessible through ODBC, but is opened exclusively in read-only mode to protect the integrity of the data.

Listed below are a few sample DSN and SQL strings for connecting to different types of databases through ODBC. Notice that they differ slightly from database to database. Remember that **Data Source** and **SQL** are expressions. If you enter a string into the expression box, the string must be properly set off by quotes. The strings in the following table appear exactly as you would type them into Lookout.

Database Type	Data Source String	SQL String

CSV	`DSN=Text Files;`
	`DBQ= c:\ldev\ald2csv;`
		`"select Priority,""Date &`
		`Time," "Description from`
		`alarm.csv where Priority = 4"`

Excel 4.0	`DSN=Excel Files;`
	`DBQ= c:\ldev\ald2csv;`
		`select Priority," "Date &`
		`Time," "Alarm Codes`
		`(Set:Reset:Acknowledge",`
		`"Description from alarm.xls`
		`alarm where Priority = 4"`

Database Type	Data Source String	SQL String
Excel 5.0	`DSN=Excel Files;`	`"select Priority," "Date & Time," "Alarm Code (Set:Reset:Acknowledged)," "Description from Table1 where Priority = 4"`
MS Access	`DSN=MS Access 7.0 Database; DBQ=c:\My Documents\compact.mdb;`	`"select Title, AlbumID, Length from tracks where TrackID = 7"`
Citadel	`DSN=Citadel 32-bit;`	`"select Interval, LocalTime," "Waves.Square" "from Traces where Interval = 0:1 and LocalTime >" "2/14/97 14:34:30"`

SQL Command Buffering

The SQLExec object can create a buffer for SQL commands so that commands are not lost if the connection to the database is temporarily lost. Buffering stores failed commands in a file so that they can be resubmitted once the connection to the database is reestablished.

If you check the **Buffer SQL** checkbox in the Create SQL dialog box, the object buffers entries that fail for the following reasons.

- Unable to connect to data source: error code 08001.
- Connection in use: error code 08002.
- Communication link failure: error code 08S01.
- Drivers SQLAllocEnv failed: error code IM004.
- Drivers SQLAllocConnect Failed: error code IM005.
- Unable to load translation DLL: error code IM009.

Entries are not buffered for any other error codes. This is to prevent entries that always cause an error from being buffered, such as a syntax error in an SQL string.

You should only enable buffering for SqlExec objects that are inserting new data into a database, such as an SQL insert command.

When entries are being buffered, they are buffered according to the data source with which they are trying to connect. This prevents an SqlExec

object being buffered for an Excel data source from blocking an SqlExec object connected to an MS Access data source.

Separating the buffers by data source also keeps entries from different SqlExec objects trying to connect to the same data source in their proper chronological order.

If an SqlExec object has to buffer data, it keeps the data in a directory called `dsndata` in your Lookout data directory. The data is kept in a `.CSV` file, you can view in Excel. Do not try to view these files while Lookout is running.

Once an object starts buffering, it buffers all subsequent entries to that data source for all objects that have buffering turned on. Because of this, an entry with a syntax or other error that should not be buffered, can be buffered. If this occurs, the buffering system discards that entry after the entry is passed to ODBC and ODBC returns an error. You are not notified when the entry is discarded.

If an object has buffering turned off, and that data source is currently being buffered, then that entry bypasses the buffer entirely. It then either succeeds or fails according to the state of the connection to the data source.

If a data source has buffered data, the buffering system automatically tries to reconnect to the data source periodically. Once it connects, it clears the buffer by periodically de-queuing the first few entries and sending them to ODBC.

During buffering, the **Failure** data member has a slightly different meaning. If it is high, then the first entry in the buffer failed and was kept in the buffer. If it is low it means that the first entry in the buffer was successfully submitted to ODBC, and objects are still being buffered until the buffer can be cleared.

If you are creating and testing a process file, you may accumulate a large amount of buffered data in the `dsndata` directory. You can safely delete this directory when Lookout is *not* running. You will lose all the buffered data, but the SqlExec objects then loads with no errors or alarms.

SqlExec Status Messages

ODBC Environment not allocated. No buffering will occur
Indicates that memory could not be allocated for the connection to ODBC. This only occurs if your system is about to run out of memory. No buffering occurs if this alarm is set. Data that has already been buffered is not lost.

ODBC(32).DLL Not loaded
Indicates that the DLL could not be loaded. No buffering can occur if this alarm is set. Data that has already been buffered, is not lost.

***Data Source Name*: Objects are being buffered for this data source**
This indicates which data sources are being buffered. Once the buffered entries have been cleared, the alarm ceases.

Incorrect data source string, check syntax
This means that the Data Source string is missing the `DSN=`*`driver name`*`;` pair.

SqlExec displays any error messages that it receives from ODBC as an alarm. If the error is an error that should cause buffering, the status message indicates whether the entry was buffered or not buffered. Each message returned by ODBC belongs to the ODBC group. Some of the more common ODBC error messages are:

[Not Buffered] 37000/-3100: [Microsoft][*Driver Name*] Syntax error in query expression ' …'
The SQL string has a syntax error in it.

[Not Buffered] S1000/-1811 [Microsoft][*Driver Name*]Couldn't find file '(unknown)'
This usually means that your data source expression is incorrect, or you have ODBC configured incorrectly for that data source expression, such as trying to connect to an Excel 5.0 workbook your ODBC Excel driver set up for Excel 4.0 worksheets. Check the data source expression and the configuration of the ODBC driver.

[Not Buffered] 42000/-1809 [Microsoft][*Driver Name*]can't update. Database or object is read-only.
The file has been opened in read-only mode, but you are trying to write to it. If you have the data member **ReadOnly** set to TRUE, set it to FALSE. For some drivers, you can specify in the ODBC administration tool that all files opened by that driver should be opened in read-only mode. Check the ODBC configuration if you have **ReadOnly** set to FALSE and are still getting this error. For some ODBC drivers (such as Citadel) the file is always opened in read-only mode to protect the integrity of the files.

SquareD

Lookout uses the SquareD object class to communicate with the SquareD family of SCP PLCs. These include the SCP-1xx, 3xx, 4xx, 5xx, 6xx, and 7xx series. Lookout can interface to Square D PLCs through either a serial interface or a SY/MAX interface. The SY/MAX interface supports both SY/MAX and net-to-net communication modes.

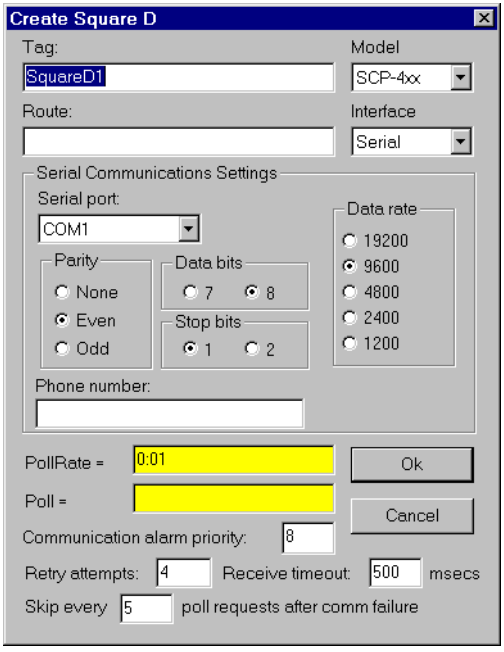


Figure 18-116. SquareD Definition Parameters Dialog Box Configured for Serial Communications

Route refers to a unique path through any network devices that leads to the PLC port. Remember, when using the SY/MAX card, the first route must be a zero followed by the address of the card. This is because the SY/MAX card edge is considered to be port zero. The RS422 port on the SY/MAX card is considered to be port one.

Model specifies the particular type of PLC you are representing with this object. The **Model** you select determines what native data members comprise the object.

PollRate is a numeric expression that determines how often to poll the device. Lookout polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *Introduction*, for information on entering time constants.

Poll is a logical expression. When this expression changes from FALSE to TRUE, Lookout polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Communication alarm priority determines the priority level of the alarms generated by the SquareD object.

Retry attempts specifies the consecutive number of times Lookout attempts to establish communications with a device if it is not getting a valid response. After it tries the number of **Retry attempts** specified, the object generates an alarm and begins to **Skip every** *n* **poll requests after comm failure**. Once Lookout reestablishes communications, it polls the device on its regular cycle, as defined by **PollRate**.

Receive timeout is the time Lookout waits for a response from a device before retrying the poll request.

Serial Port Interface Parameters

The **Serial Interface** selection enables serial port communication from your computer to the programming port on your PLC. The previous diagram shows a Square D PLC configured for serial communications.

Serial port specifies which RS-232C port on your computer the object uses for communication to the physical device.

Data rate, **Parity**, **Data Bits**, and **Stop bits** reference the settings of the hardware device. Choose the settings as configured on you PLC.

Phone number specifies the number to be dialed if the serial port setting is configured for dial-up. This number only applies to the individual object. See **Options»Serial Ports** for more information.

SY/LINK Interface Parameters

The `SY/LINK` **Interface** selection enables direct connection of your computer to a Square D network using a `SY/MAX` card. The diagram below shows a SquareD object configured for `SY/MAX` communications.

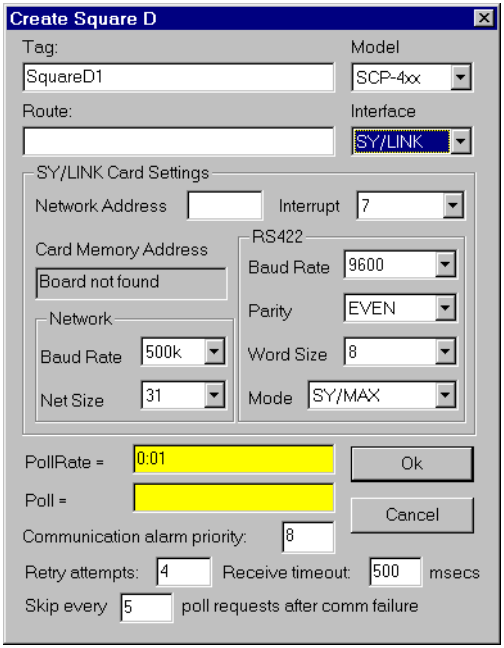


Figure 18-117. SquareD Definition Dialog Box Configured for SY/LINK Communications

Network address identifies the address of your computer interface card in the SY/MAX network. Valid addresses range from 0 to 99. The card node address must be unique—that is, it must not be the same as the address of any other device on the network.

Interrupt identifies the interrupt (IRQ) setting of your SY/MAX interface card in your PC. Assigning an interrupt to the interface card improves overall computer performance. Any time the card receives an response, it generates an interrupt recognized by Lookout.

Card memory address specifies the base address location of the card memory. This is selected for you automatically. At present, only one SY/MAX card in a computer is supported. Lookout looks for the card and fills this box with its address. If the card is not found, this is indicated here as well.

Network settings include net size and baud rate. **Net size** identifies the number of devices on the network. The SY/MAX card and all of the **PLCs** or **NIMs** must have the same settings for successful communications to occur. A net size of 31 should promote a faster response time. **Baud rate**

selects the baud rate at which the SY/LINK card tries to communicate on the SY/NET network.

The **RS-422** settings include **baud rate**, **parity**, and **word size**. These parameters reference the settings of the hardware device. Choose the settings as configured on you PLC.

RS-422 mode chooses either SY/MAX or Net-to-net. Choosing SY/MAX allows normal network operations. Choosing Net-to-net allows extended distributed networks, large capacity networks (more than 200 devices), or network redundancy (more than one path between two devices). The default is SY/MAX.

SY/ENET Interface Parameters

The SY/ENET **Interface** selection enables direct connection of your computer to a Ethernet network using an Ethernet card. The diagram below shows a SquareD object configured for SY/ENET communications.

To use SquareD with the ENET protocol, you must install the Lookout Ethernet driver as a protocol in the Windows Network Control Panel. The files LKETHER.SYS and OEMSETUP.INF must be installed when Lookout is installed.

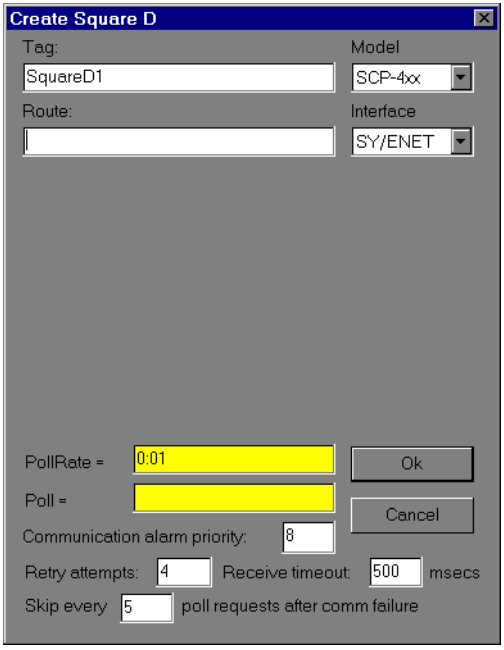


Figure 18-118. SquareD Definition Dialog Box Configured for SY/ENET Communications

Route refers to a unique path through any network devices that leads to the PLC port.

SquareD Data Members

Each SquareD object contains a great deal of data. All readable and writable members (inputs/outputs) are bundled with the object. As soon as you create an object you immediately have access to all the object data members.

☞ **Note** ***The SquareD object class automatically generates an efficient read/write blocking scheme based on the inputs and outputs you are using in your process file. You are not required to build your own I/O blocking table. However, you can ensure peak performance by organizing your PLCs data into contiguous groups.***

Table 18-124. SquareD Data Members

Data Member	Type	Read	Write	Description
1 – 8176	numeric	yes	yes	16-bit unsigned integer ranging from 0 to 65535
1. – 8176.	logical	yes	yes	Logical I/O that reads and writes the entire register. When you write to the register, all bits in the word go TRUE or FALSE.
1.0 – 8176.15	logical	yes	yes	Individual bits in a register read as logical ON/OFF values. The least significant bit is 0; the most significant, 15.
CommFail	logical	yes	no	Object-generated signal that is on if, for whatever reason, Lookout cannot communicate with the PLC.
F1 – F8175 (odd only)	numeric	yes	yes	32-bit IEEE floating point register—reads two adjacent holding registers as a single 32-bit floating point value
Poll	logical	no	yes	When this value transitions from FALSE to TRUE, the Lookout object polls the PLC device
PollRate	numeric	no	yes	Specifies the frequency at which the Lookout object polls the PLC device
S1 – S8176	numeric	yes	yes	16-bit signed word ranging from –32,768 to +32,768
Update	logical	yes	no	Object-generated signal that pulses each time the object polls the device

SquareD Error Messages

The SquareD object class reports the status of commands it issues to the PLC. When Lookout receives an error response from a PLC, it reports the failure as an alarm containing the status code and its meaning. The following are a few examples of such alarms:

SY/MAX initialization error:(0)Cannot initialize SY/LINK board
SY/MAX polling error:(3)Illegal address attempted
Response error:(3) Received NAK in response

Switch

Switch generates a logical signal for receipt by other objects. Switches change state when you click on them with a mouse button, trackball, touchscreen, or space bar on your keyboard.

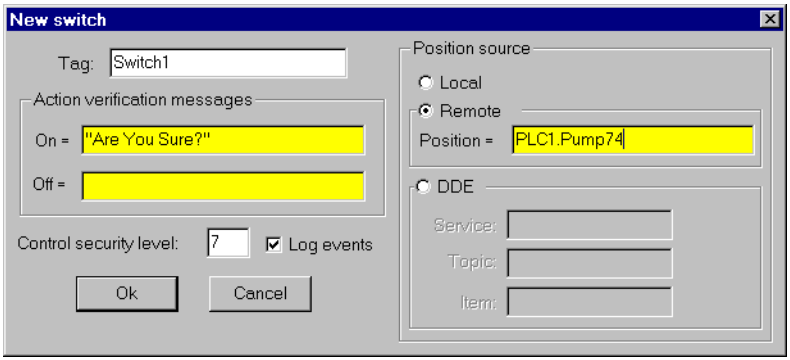


Figure 18-119. Switch Definition Parameters Dialog Box

Use **Action verification messages** to create dynamic text expressions to be displayed in message dialog boxes. See Chapter 10, *Security*, for more information on security.

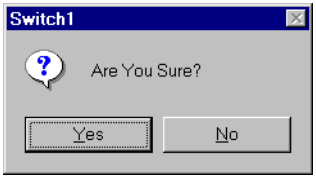


Figure 18-120. Verification Message Dialog Box

Position source determines where the value of the switch resides. **Local** indicates the value of the switch lies within the object itself—on the control panel. If the switch is up the signal is ON, if down the signal is OFF.

Remote switches get their values from a remote source, often the register on a controller they are connected to. Flipping the switch changes the status of the register, and changing the status of the register flips the switch.

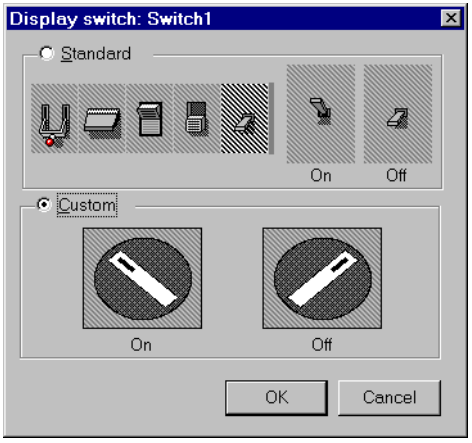
The **Remote** option is especially useful when you want to prevent Lookout from changing the value of setpoints or registers upon initial startup, or reconnection of lost communication. When you use this style switch, you are creating a sort of looped signal. Half the loop is formed when you

connect the controller register to the switch with the **Position** expression, while the second half is formed when you connect the switch output signal to the controller register. Notice **Position** is a logical expression. Do not forget to complete the second half of the loop with the **Object»Edit Connections…** command.

Much like Remote switches, **DDE** (Dynamic Data Exchange) switches get their values from a remote source. This could be a cell in a spreadsheet, another DDE aware application, or a second copy of Lookout running on the network. See Chapter 13, *Dynamic Data Exchange*, for more information on **Service**, **Topic**, and **Item** parameters.

Control security level specifies the minimum security level operators must have to gain access to this individual object, and thus control it.

The **Log events** option creates a permanent audit trail for the object—who did what and when. All adjustments of the switch are logged to disk, including the time the switch was flipped, the operator's account name, and the direction the switch was flipped. See Chapter 11, *Logging Data and Events*, for more information on event logging.



You can replace the standard switch types with custom graphic symbols. If you decide to use custom graphics, you must specify both symbol parameters, **On** and **Off**. See Chapter 8, *Graphics*, for more information on creating custom graphic symbols and the use of Transparent pixels.

Switch Data Members

Table 18-125. Switch Data Members

Data Members	Type	Read	Write	Description
(implicit)	logical	yes	no	Switch Position
enable	logical	no	yes	If TRUE (the default), enables DDE. If FALSE, disables DDE. The default value is ON. This input is ignored for non-DDE TextEntry objects.
visible	logical	no	yes	When FALSE, the switch object cannot be seen on the display panel. When TRUE, the switch can be seen and controlled.

Comments ***If a switch with more than two positions is needed, use a Pot object instead. See Chapter 5,*** *Developer Tour,* ***for an example of a multiposition switch.***

Related Objects ***Pushbutton, Pot***

$System

$System is a global object. It makes global Lookout data such as the currently logged in user name and security level available for use in your process. You can use $System data members just like other object data members.

The **seclevel** data member is always an integer value between 1 and 10. This number represents the Lookout security level of the user currently logged in. For more information about Lookout security levels, see Chapter 10, *Security*.

The **time** data member represents the current date and time of the system. Like all time values in Lookout, this is a floating point number in which the integer represents the date and the fraction represents the time of day. You can use the various Lookout date and time numeric formats to view this value in the most convenient format. This data member updates itself every minute, on the minute. It also updates itself immediately after it is created or when its process is opened.

The **username** data member is the account name of the user currently logged in. For more information about Lookout security accounts, see Chapter 10, *Security*.

System Data Members

Table 18-126. $System Data Members

Data Member	Type	Read	Write	Description
seclevel	numeric	yes	no	Security level of the user currently logged in
time	numeric	yes	no	Current operating system time
username	text	yes	no	Name of the user currently logged in

Tesco

Tesco is a protocol driver object class Lookout uses to communicate with Liquitronic LIQ programmable controllers using the **LIQ4** (Data Express) and **LIQ5** (Data Express Plus) messaging protocols. Create one Tesco object for each controller.

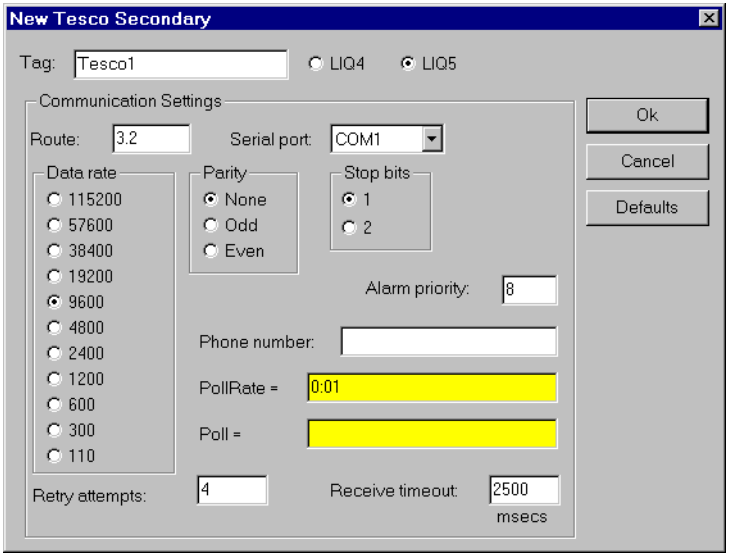


Figure 18-121. Tesco Configuration Parameters Dialog Box

Route refers to the PLC address setting as specified in its Configuration Table. When using **LIQ4**, it is a simple node address (1 to 255). When using **LIQ5**, Route contains both the network address and the node address in the format `network.node`, as shown in the diagram.

Serial port specifies which port the object uses for communication to the external device. This does not specify the communication topology (such as, radio, dial-up, hard wired). Communication type is determined by the **Options»Serial Ports…** command.

Data rate, **Parity**, and **Stop bits** reference the settings on the hardware device.

The **Defaults** button replaces the current settings with default values.

Alarm priority determines the priority level of Tesco communication alarms.

Phone number specifies the number to be dialed if the serial port setting is configured for dial-up. This number only applies to the individual driver object.

PollRate is a numeric expression that determines how often to poll the device. The Tesco object then polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *Introduction*, for more information on entering time constants.

Poll is a logical expression. When this expression changes from FALSE to TRUE, Lookout polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Retry attempts specifies the consecutive number of times Lookout attempts to establish communications with a device it does not get a valid response from. After **Retry attempts** times, the object generates a communication alarm and Lookout moves on to the next device in the polling queue (if any).

Receive timeout is the time delay Lookout uses in waiting for a response from a device before retrying the request.

Tesco Data Members

Like other protocol driver objects, Tesco objects can contain a great deal of data. All readable and writable members (inputs/outputs), polling instructions, and so on, are bundled with the object. As soon as you create a Tesco object you immediately have access to all the object data members (see the data member list below).

Table 18-127. Tesco Data Members

Data Members	Type	Read	Write	Description
AO0 – AO1023	numeric	yes	yes	Analog Output register capable of holding a 32-bit IEEE floating point value for use in internal Tescode programming, or a physical AO signal value as a 12-bit whole number ranging from 0 to 4095.
CommFail	logical	yes	no	Driver-generated signal that is ON if Lookout cannot communicate with the device for whatever reason
IR0 – 1R1023	numeric	yes	yes	Index register containing a 32-bit unsigned integer ranging from 0 to 4,294,967,295.
L0 – L1023	numeric	yes	yes	Level (Analog Input) register capable of holding a 32-bit IEEE floating point value for use in internal Tescode programming, or a physical AI signal as a 12-bit whole number ranging from 0 to 4095.
P0 – P1023	logical	yes	yes	Pump (Discrete Output) register associated with a physical output channel.
Poll	logical	no	yes	When this value transitions from FALSE to TRUE, Lookout polls the device.
PollRate	numeric	no	yes	Specifies the frequency at which the device is to be polled.
S0 – S1023	logical	yes	yes	Status (Digital Input) register that can represent either a physical input channel or an internal Tescode programming flag.
SP0 – SP1023	numeric	yes	yes	Setpoint register that holds a 32-bit IEEE floating point value with a range of +/–3.37 × 1038

Table 18-127. Tesco Data Members (Continued)

Data Members	Type	Read	Write	Description
T0 – T1023	numeric	yes	yes	Timer/Counter register containing Pulse counters, Hours timers, HMS timers, Event counters, and Seconds timers as documented in the TESCO LIQ 5 Programmable Control Operations Manual
Update	logical	yes	no	Driver-generated signal that pulses each time the driver polls the device

TextEntry

With TextEntry you can manually enter textual notes with the keyboard. These notes may contain any combination of numeric and alphanumeric characters; however, the result of your entry is converted to a text value. Just like any other text expression in Lookout, your note can be logged to disk, connected to other data members that accept text signals, and so on. The note is saved and displayed as a single line entry—you cannot embed carriage returns into the message.

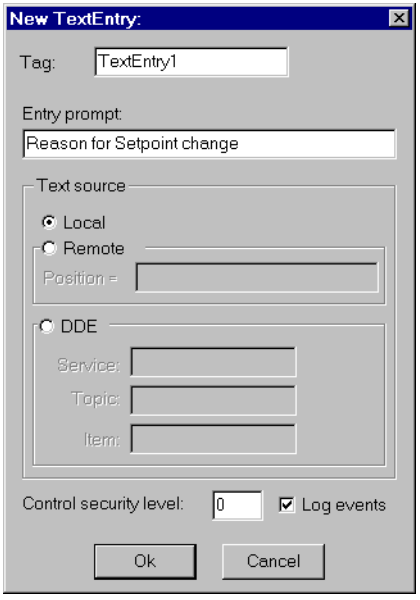


Figure 18-122. TextEntry Parameters Dialog Box

Entry prompt is the text that appears at the top of the text entry dialog box when an operator selects the text entry pushbutton.

Text source determines where the user-entered text resides. **Local** indicates the user-entered text lies within the object itself—on the control panel.

Remote indicates that the user-entered text resides in a remote source, such as a text expression or another TextEntry object.

Much like **Remote** TextEntry objects, **DDE** TextEntry objects get their values from a remote source. This is the option you use to tie the text to a cell in a spreadsheet, a database lookup table, or any DDE aware

application—including a second copy of Lookout running on the network. See Chapter 13, *Dynamic Data Exchange*, for more detailed information on **Service**, **Topic** and **Item**.

☞ **Note** ***The last DDE parameters used on any object automatically become the default values for any new DDE object.***

Control security level specifies the minimum security level operators must have to gain access to this individual object, and thus control it.

The **Log events** option creates a permanent audit trail for the object—who did what and when. When selected, all text entries in this object are logged to disk. Each entry includes the time of the entry, the operator's account name, and what entry was made. See Chapter 11, *Logging Data and Events*, for more information on event logging.

Lookout presents the following display parameters dialog box after you define the object. It lets you define the text font and presentation style.

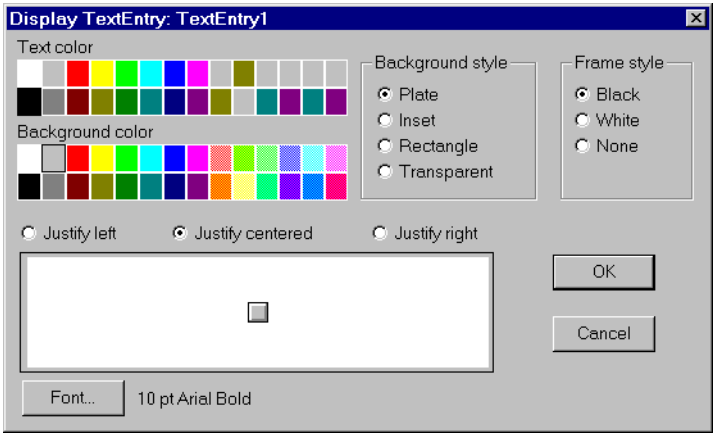


Figure 18-123. TextEntry Display Parameters Dialog Box

TextEntry Data Members

Table 18-128. TextEntry Data Members

Data Members	Type	Read	Write	Description
(implicit)	text	yes	no	Current, user-entered text
enable	logical	no	yes	If TRUE (the default), enables DDE. If FALSE, disables DDE. The default value is ON. This input is ignored for non-DDE TextEntry objects.

TimeOf*xxxx*

TimeOf*xxxx* are timers that generate a periodic pulse of a specified duration. The timers are turned on and off by **On/off signal**. The time period is defined by the type of timer used—a TimeOfMinute timer has a one-minute period, a TimeOfYear timer has a one-year time period, and so on. The output of these timers goes high after the specified **Timer offset** has elapsed in the current period and remains high for the specified **Timer duration**.

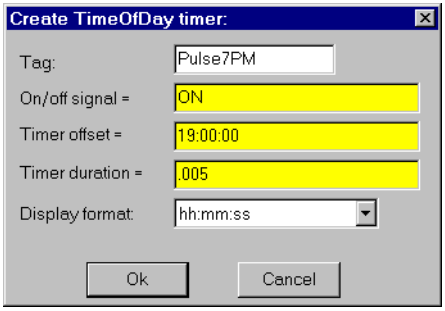


Figure 18-124. TimeOfDay Definition Parameters Dialog Box

The **Timer offset** and **Timer duration** can range from 0.0 seconds to a year, and the effective resolution is 0.01 seconds over the entire range. The **Timer offset** plus the **Timer duration** should always be less than or equal to the time period.

The object display shows the time remaining before the output changes state and is updated approximately once per second. It is shown in the selected **Display format**. If the **On/off signal** is OFF, the display shows OFF.

The **On/off signal** is a logical expression while **Timer offset** and **Timer duration** are numeric expressions. Normally, these are simple time constants such as 6:10:20 (six hours:ten minutes:twenty seconds). See *Numeric Data Members* in Chapter 5, *Developer Tour*, for more information on entering time constants.

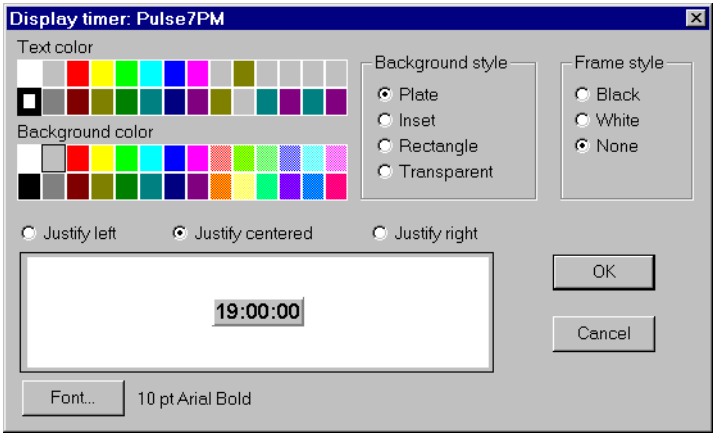


Figure 18-125. TimeOfDay Display Parameters Dialog Box

Timeofxxxx Data Members

Table 18-129. TimeOfxxxx Data Members

Data Members	Type	Read	Write	Description
(implicit)	logical	yes	no	Logical timer value

Comments ***TimeOfxxxx can be used in place of Pulse objects when the pulse needs to be synchronized with the clock—if a pump should only be allowed to run between the hours of 8:00 and 17:00 each day, the TimeOfDay timer should be used.***

Related Objects ***DelayOff, DelayOn, Interval, OneShot, Pulse***

Tiway

Tiway is a protocol driver object Lookout uses to communicate with series 5*xx* PLCs manufactured by Siemens, formerly made by Texas Instruments.

Protocol driver objects contain a great deal of data. All readable and writable members (inputs/outputs), polling instructions, read/write blocking, serial port usage, and so on are bundled with the object. As soon as you create a Tiway object you immediately have access to all the object data members (see data member list below).

☞ **Note** ***Lookout protocol driver objects automatically generate an efficient read/write blocking scheme based on the inputs and outputs being used in your process file. You are not required to build your own I/O blocking table.***

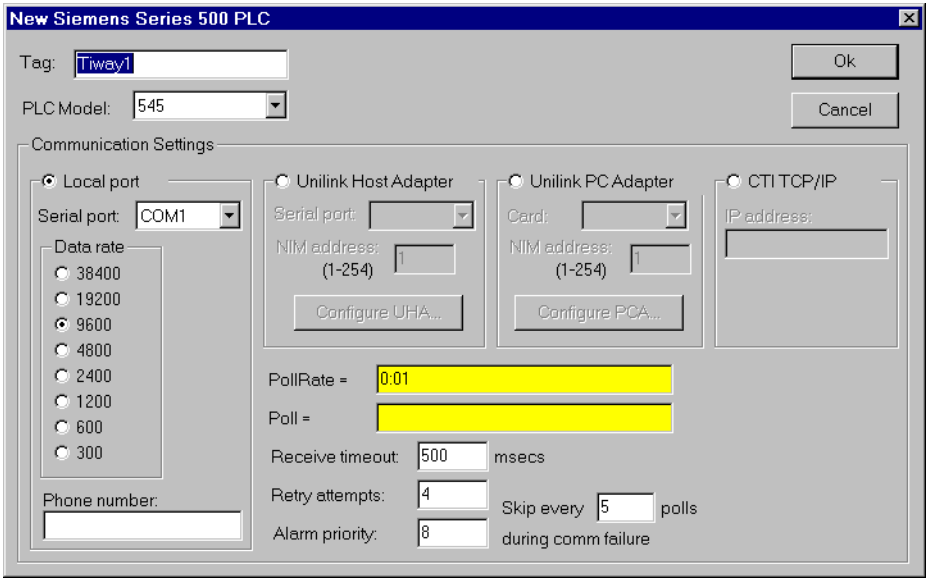


Figure 18-126. Tiway Configuration Parameters Dialog Box

PLC Model specifies the PLC model number for the requested device.

PollRate is a numeric expression that determines how often to poll the device. Tiway then polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *Introduction*, for more information on entering time constants.

Poll is a logical expression. When this expression changes from FALSE to TRUE, Lookout polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Receive timeout is the time delay Lookout uses in waiting for a response from a device before retrying the request.

Retry attempts specifies the consecutive number of times Lookout attempts to establish communications with a device it does not get a valid response from. After Retry attempts times, Tiway generates a communication alarm and Lookout moves on to the next device in the polling queue (if any).

Alarm priority determines the priority level of Tiway generated alarms.

Phone number specifies the number to be dialed if the serial port setting is configured for dial-up. This number only applies to the individual protocol object.

The **Skip every ___ polls** setting instructs Lookout not to poll a device it has lost communication with on every scheduled poll. Instead, Lookout skips the device in the polling cycle accordingly. Once communications have been reestablished, the device is polled on its regular cycle.

Communication Techniques

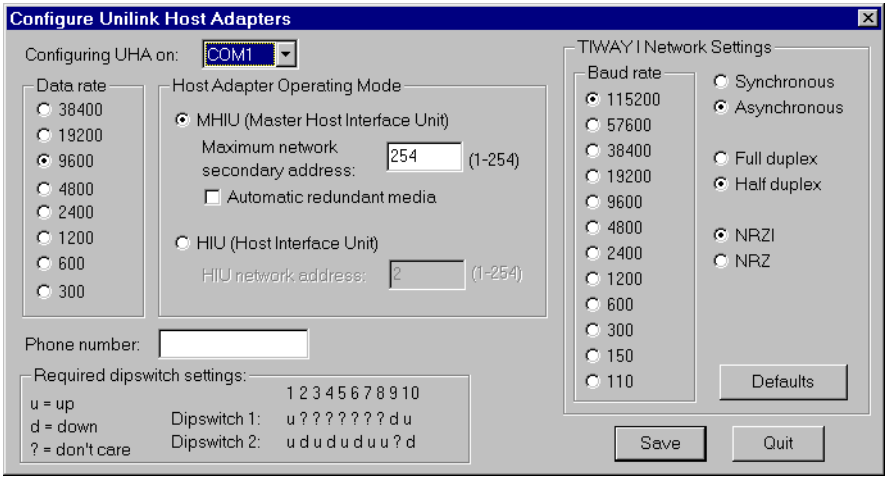
Lookout communicates with Siemens PLCs in several ways: direct serial connection to the **Local port**, serial connection to an **external Unilink Host Adapter**, through an internal **Unilink PC Adapter** card, or through an internal **CTI TCP/IP** card.

Local Port

The **Local port** settings determine the **serial port**, **data rate**, and **phone number** (if any) to be used in a direct connect setup. Because the **Local port** protocol does not include address information, this option is limited to only one (1) PLC per serial port.

Unilink Host Adapter

If **Unilink Host Adapter** is selected, you must specify the **Serial port** to be used and the **NIM** (Network Interface Module) **address** as set at the PLC. You also should configure several settings on the Unilink Host Adapter by selecting the **Configure UHA…** button.



The settings in this dialog box are globally applied to all PLCs on the specified TIWAY network (each network requires a separate serial port). Therefore, it is only necessary to configure each Unilink Host Adapter one time— you need not repeat this step every time you create a new Tiway object.

Data rate specifies the communication speed between the computer and the Unilink Host Adapter. It also determines the required dip switch settings on the UHA for the selected baud rate.

The **Host Adapter Operating Mode** determines if the Unilink Host Adapter is the network manager (**Master Host Interface Unit**) or just another network secondary (**Host Interface Unit**). There must be exactly one MHIU per TIWAY network.

Enabling **Automatic redundant media** instructs the Unilink Host Adapter to attempt communications over a redundant TIWAY network to any secondary it loses communications with.

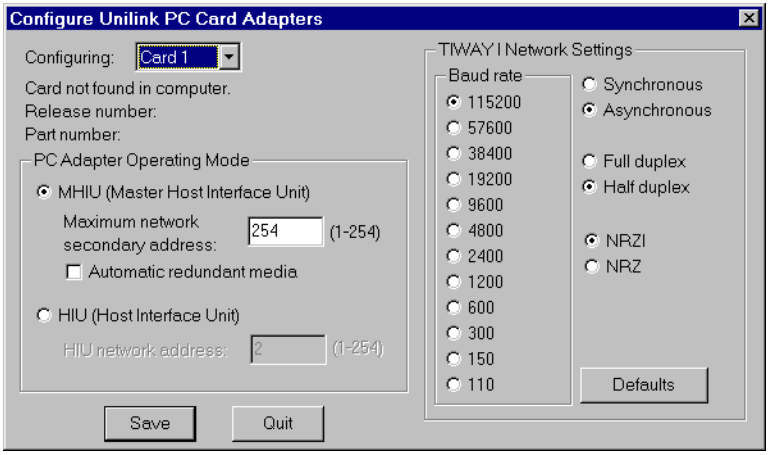
The **TIWAY I Network Settings** configure the communication parameters for the TIWAY network. This network runs between the Unilink Host Adapter and its secondaries. The Lookout default network settings correspond to the default NIM settings as shipped from Siemens. See your TIWAY documentation to modify any of these parameters.

Unilink PC Adapter

Because the **Unilink PC Adapter** is an internal card, it eliminates the 19200 baud serial bottleneck and replaces it with the 8 MHz PC ISA bus

speed. Therefore, the performance gains over the **Unilink Host Adapter** and **Local port** settings can be substantial.

If **Unilink PC Adapter** is selected, you must specify the **Card** to be used and the **NIM** (Network Interface Module) **address** as set at the PLC. You should also configure several settings on the Unilink PC Adapter by selecting the **Configure PCA…** button.



The settings in this dialog box are globally applied to all PLCs on the specified TIWAY network (each network requires a separate card). Therefore, it is only necessary to configure each Unilink PC Adapter one time—this step need not be repeated every time a new Tiway object is created.

The **PC Adapter Operating Mode** determines if the Unilink PC Adapter is the network manager (**Master Host Interface Unit**) or just another network secondary (**Host Interface Unit**). There must be exactly one MHIU per TIWAY network.

Enabling **Automatic redundant media** has no effect with the PC Adapter card because it has only one port. If Siemens adds a second port, Lookout automatically supports this option.

The **TIWAY I Network Settings** configure the communication parameters for the TIWAY network. This network runs between the Unilink PC Adapter card and its secondaries. Lookout default network settings correspond to the default NIM settings as shipped from Siemens. See your TIWAY documentation to modify any of these parameters.

CTI TCP/IP

Lookout supports the Control Technology Incorporated (CTI) Ethernet TCP/IP adapter cards that can be installed in SIMATIC TI545 PLCs. In order to work with such cards, your PC must be equipped with an Ethernet network card and a Windows Sockets-Compliant TCP/IP software package. Such packages are available from Microsoft, FTP Software, and NetManage, Inc.

The Lookout **CTI TCP/IP** protocol option is Windows Sockets Compliant. It uses connectionless UDP sockets in software, an industry standard for TCP/IP protocols. In this protocol, a FIFO (first-in, first-out) stack is used to temporarily store communication messages if the data highway is busy or if multiple poll request are generated by several Tiway objects.

Because **CTI TCP/IP** utilizes sockets to momentarily store poll requests, this protocol eliminates bottlenecks imposed by multiple Tiway objects trying to access the data highway at the same time. Performance gains over **Local port**, **Unilink Host Adapter** and **Unilink PC Adapter** settings can be substantial when you are configuring a system that has several PLCs on the same network.

If **CTI TCP/IP** is selected, you need to specify the **IP address** (Internet protocol address) of the PLC. An Internet protocol address consists of four numbers, separated by periods. Each number ranges from zero to 255 decimal. Thus, a typical Internet address might be 128.7.9.231. Ensure that the **IP address** you enter matches the Internet protocol address of the PLC as specified in its EEPROM or as programmed using PCL.

You can add a secondary IP address to the **CTI TCP/IP** parameter. Lookout now toggles between the primary and secondary IP address after a COM failure (assuming a secondary address exists). Enter the secondary ID after the first, preceded by a space or a comma. For example:

```
207.68.156.61, 1.2.3.4
```

Tiway Data Members

Table 18-130. Tiway Data Members

Data Members	Type	Read	Write	Description
STW1 – STW32000	numeric	yes	no	Status Words
X1 – X32000	logical	yes	no	Discrete Inputs—unassigned Xs may be used as control registers

Table 18-130. Tiway Data Members (Continued)

Data Members	Type	Read	Write	Description
Y1 – Y32000	logical	yes	yes	Discrete Outputs—same memory space as Discrete Inputs, so X37 references the same point as Y37. Unassigned Ys may be used as control registers
C1 – C32000	logical	yes	yes	Control Registers
V1T – V32000T	text	yes	yes	Two characters of text
V1B1 – V32000B16	logical	yes	yes	One bit of a word written out as a whole word
WX1 – WX32000	numeric	yes	no	Word Image Inputs—16-bit values that typically range from 6400 – 32000 for 4 – 20 mA signals, and 0 – 32000 for 0 – 5V signals.
WY1 – WY32000	numeric	yes	yes	Word Image Outputs—16-bit values that typically range from 6400 – 32000 for 4 –20 mA signals, and 0 – 32000 for 0 – 5V signals.
V1 – V32000	numeric	yes	yes	V-memory unsigned 16-bit integer value ranging from 0 to 65535
V1S – V32000S	numeric	yes	yes	V-memory signed 16-bit integer value ranging from –32768 to 32767
V1. – V32000.	numeric	yes	yes	V-memory 32-bit IEEE floating point value
V1D – V32000D	numeric	yes	yes	V-memory 32-bit unsigned integer value
K1 – K32000	numeric	yes	yes	K-memory unsigned 16-bit integer value ranging from 0 to 65535
K1S – K32000S	numeric	yes	yes	K-memory signed 16-bit integer value ranging from –32768 to 32767
K1. – K32000.	numeric	yes	yes	K-memory 32-bit IEEE floating point value
K1D – K32000D	numeric	yes	yes	K-memory 32-bit unsigned integer value

Table 18-130. Tiway Data Members (Continued)

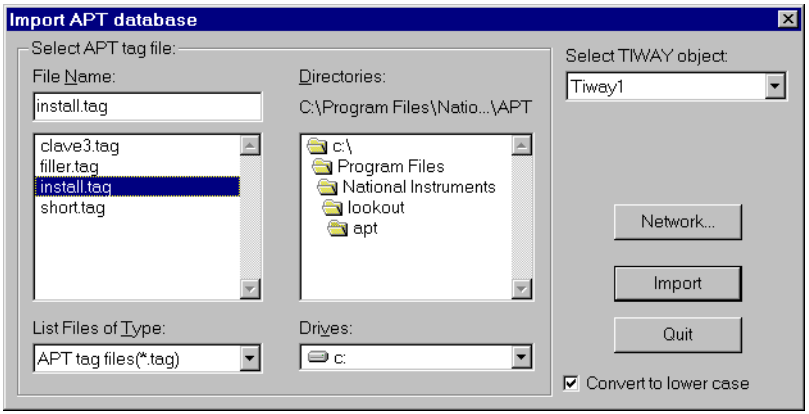
Data Members	Type	Read	Write	Description
LADB1 – LADB64	numeric	yes	yes	(Analog Alarm) Deadband
AERR1 – AERR32000	numeric	yes	yes	(Analog Alarm) Error
AHA1 – AHA32000	numeric	yes	yes	(Analog Alarm) High alarm limit
AHHA1 – AHHA32000	numeric	yes	yes	(Analog Alarm) High high alarm limit
ALA1 – ALA32000	numeric	yes	yes	(Analog Alarm) Low alarm limit
ALLA1 – ALLA32000	numeric	yes	yes	(Analog Alarm) Low low alarm limit
AODA1 – AODA32000	numeric	yes	yes	(Analog Alarm) Orange deviation limit
APVH1 – APVH32000	numeric	yes	yes	(Analog Alarm) Process variable high limit
APVL1 – APVL32000	numeric	yes	yes	(Analog Alarm) Process variable low limit
ARCA1 – ARCA32000	numeric	yes	yes	(Analog Alarm) Rate of change limit
ASP1 – ASP32000	numeric	yes	yes	(Analog Alarm) Setpoint
ASPH1 – ASPH128	numeric	yes	yes	(Analog Alarm) Setpoint high limit
ASPL1 – ASPL128	numeric	yes	yes	(Analog Alarm) Setpoint low limit
ATS1 – ATS32000	numeric	yes	yes	(Analog Alarm) Sample rate
AVF1 – AVF128	numeric	yes	yes	(Analog Alarm) Alarm flags
LYDA1 – LYDA64	numeric	yes	yes	(Analog Alarm) Yellow deviation limit
TCP1 – TCP32000	numeric	yes	yes	(Analog Alarm) Timer/counter preset
TCC1 – TCC32000	numeric	yes	yes	(Analog Alarm) Timer/counter current
LMX1 – LMX64	numeric	yes	yes	(Loop) Bias
LADB1 – LADB64	numeric	yes	yes	(Loop) Deadband
LKD1 – LKD64	numeric	yes	yes	(Loop) Derivative gain
LER1 – LER64	numeric	yes	no	(Loop) Error

Table 18-130. Tiway Data Members (Continued)

Data Members	Type	Read	Write	Description
LKC1 – LKC64	numeric	yes	yes	(Loop) Gain
LHHA1 – LHHA64	numeric	yes	yes	(Loop) High high alarm limit
LHA1 – LHA64	numeric	yes	yes	(Loop) High alarm limit
LLLA1 – LLLA64	numeric	yes	yes	(Loop) Low low alarm limit
LLA1 – LLA64	numeric	yes	yes	(Loop) Low alarm limit
LODA1 – LODA64	numeric	yes	yes	(Loop) Orange deviation limit
LMN1 – LMN64	numeric	yes	yes	(Loop) Output
LPV1 – LPV64	numeric	yes	yes	(Loop) Process variable
LPVH1 – LPVH64	numeric	yes	yes	(Loop) Process variable high limit
LPVL1 – LPVL64	numeric	yes	yes	(Loop) Process variable low limit
LRCA1 – LRCA64	numeric	yes	yes	(Loop) Rate of change limit
LSP1 – LSP64	numeric	yes	yes	(Loop) Setpoint
LSPH1 – LSPH64	numeric	yes	yes	(Loop) Setpoint high limit
LSPL1 – LSPL64	numeric	yes	yes	(Loop) Setpoint low limit
LTD1 – LTD64	numeric	yes	yes	(Loop) Rate
LTI1 – LTI64	numeric	yes	yes	(Loop) Reset
LTS1 – LTS64	numeric	yes	yes	(Loop) Sample rate
LYDA1 – LYDA64	numeric	yes	yes	(Loop) Yellow deviation limit
CommFail	logical	yes	no	Driver-generated signal that is ON if Lookout cannot communicate with the device for whatever reason
Poll	logical	no	yes	When this value transitions from FALSE to TRUE, Lookout polls the device
PollRate	numeric	no	yes	Specifies the frequency at which the Lookout object polls the device
Update	logical	yes	no	Driver-generated signal that pulses each time the driver polls the device

Importing APT Tag Files

After you have created at least one Tiway object, the Tiway class adds a menu selection to the Lookout **Options** menu you can use to import an APT tag file database for each Tiway object created. You can re-import tag files as your APT programs are modified, and Lookout readjusts the aliased tag names automatically, in real time.



Toshiba Mseries/Toshiba Tseries

Toshiba is a protocol driver class Lookout uses to communicate with Toshiba M Series Ex100, M20, M49 and T Series T1, T2, and T3 devices using the Host Link serial communication protocol.

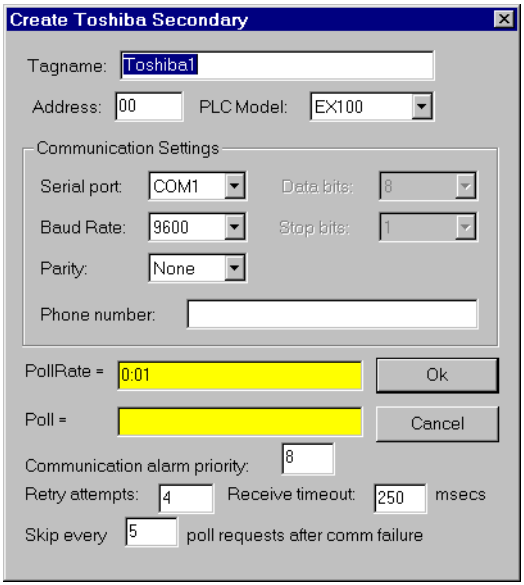


Figure 18-127. Toshiba M Series Configuration Parameters Dialog Box

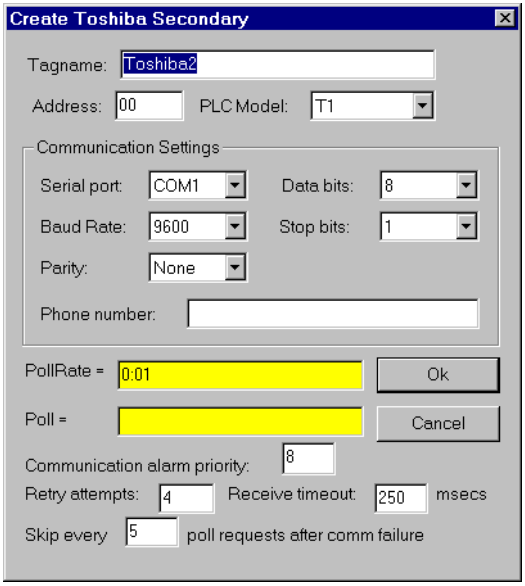


Figure 18-128. Toshiba T Series Configuration Parameters Dialog Box

Address specifies the address of the PLC. The maximum address for a T Series PLC is 32 and for an M Series PLC, 15.

Serial port specifies which COM port the object uses for communicating to the external device. This does not specify the communication type. Communication type is determined by the **Options»Serial Ports…** command.

Data rate indicates the baud rate that Lookout uses to communicate with the hardware device. The **Data rate** setting should match the selection made on the physical device.

Data bits indicates the number of data bits that Lookout uses to communicate with the hardware device. The **Data bits** setting should match the selection made on the physical device.

Stop bits indicates the number of stop bits that Lookout uses to communicate with the hardware device. This setting should match the selection made on the physical device.

Parity indicates the parity that Lookout uses to communicate with the hardware device. This **Parity** setting should match the selection made on the physical device.

Phone number specifies the number to be dialed if the selected serial port is configured for dial-up. This **Phone number** only applies to the individual protocol object.

PollRate is a numeric expression that determines how often to poll the device. The object then polls the device at the specified time interval. Normally, this is a simple time constant such as 0:01 (one second). See *Numeric Data Members* in Chapter 2, *Introduction*, for more information on entering time constants.

Poll is a logical expression. When this expression changes from FALSE to TRUE, Lookout polls the device. You can use a simple expression like the signal from a pushbutton, or a complex algorithm.

Communication alarm priority determines the priority level of alarms generated by the Toshiba object. Such alarms are typically related to communications with the physical device.

Retry attempts specifies the consecutive number of times Lookout attempts to establish communications with a device when it is not getting a valid response. After it tries the number of **Retry attempts** specified, the Toshiba object generates an alarm and releases the communication port back to the communications subsystem. The subsystem then moves on to the next device in the polling queue (if any). See Chapter 6, *Serial Communications*, for more information.

Receive timeout is the time delay Lookout uses in waiting for a response from a device before retrying the request.

The **Skip every…** setting instructs Lookout not to poll a device it has lost communication with on every scheduled poll. Instead, Lookout skips the device in the polling cycle. Once communications have been reestablished, the device is polled on its regular cycle.

Toshiba Data Members

A Toshiba object contains a great deal of data. You can read and write to all predefined data points. When you create a Toshiba object, you have immediate access to all the data members for that object.

As with all Lookout drivers, you can access I/O points and other data through data members. The following is a table of data members currently available in the Toshiba object class.

Table 18-131. Toshiba M Series Data Members

Data Member	Type	Read	Write	Description
CommFail	logical	yes	no	Object-generated signal that is ON if, for any reason, Lookout cannot communicate with the device(s).
D:0, D:1536	numeric	yes	yes	Data register.
R:0, R:1024	logical	yes	yes	Auxiliary relay device.
RW:0, RW:64	numeric	yes	yes	Auxiliary relay register.
T:0, T:128	numeric	yes	yes	Timer register.
Y:0, Y:512	logical	yes	yes	External output device.
Poll	logical	no	yes	When this value transitions from FALSE to TRUE Lookout polls the device.
PollRate	numeric	no	yes	Lookout expression that determines the device polling frequency.
Update	logical	yes	no	Object-generated signal that pulses low each time it polls the device.
YW:0, YW:64	numeric	yes	yes	External output register.
Z:0, Z:512	logical	yes	yes	Link device.
ZW:0, ZW:64	numeric	yes	yes	Link register.
C:0 – C:1536	numeric	yes	no	Counter.

Table 18-132. Toshiba T Series Data Members

Data Member	Type	Read	Write	Description
CommFail	logical	yes	no	Object-generated signal that is ON if, for any reason, Lookout cannot communicate with the device(s).
D:0, D:1536	numeric	yes	yes	Data register.
Poll	logical	no	yes	When this value transitions from FALSE to TRUE Lookout polls the device.

Table 18-132. Toshiba T Series Data Members (Continued)

Data Member	Type	Read	Write	Description
PollRate	numeric	no	yes	Lookout expression that determines the device polling frequency.
R:0, R:1024	logical	yes	yes	Auxiliary relay device.
RW:0, RW:64	numeric	yes	yes	Auxiliary relay register.
S:0 – S:1024	logical	yes	yes	Binary timer register.
SW:0 – SW:62	numeric	yes	yes	Data register.
T:0, T:128	numeric	yes	no	Timer register.
Update	logical	yes	no	Object-generated signal that pulses low each time it polls the device.
X:0, X:512	logical	yes	no	External input device.
XW:0, XW:64	numeric	yes	no	External input register.
Y:0, Y:512	logical	yes	yes	External output device.
YW:0, YW:64	numeric	yes	yes	External output register.

Toshiba Status Messages

No response within timeout period
Lookout received no response from a device within the **Receive timeout** period. The Toshiba object is able to use the COM port, but when it polls the device, it does not respond—as if it is not even there.

Toshiba errors reported in the response
These errors are reported by the Toshiba device and are in turn reported to the user in text form.

Missing address marker in frame.
Invalid address in response.
Invalid command in response.
Missing BCC marker in frame.
Invalid BCC.
Missing end of frame marker.
All these alarms indicate a garbled response frame. Check the receive gap or the retry setting in Lookout.

Trend

A Trend object displays a real-time trend graph on a control panel with any number of logical and numeric trend lines. Its graph scrolls from right to left, with current signal levels at the right end of the graph and the oldest values scrolling off the left end of the graph.

The Trend object cannot plot data from the Citadel historical database. Therefore, you cannot view data that scrolls off the left end of the chart. In contrast, the HyperTrend object class has access to information stored in the Citadel database, enabling you to view both real-time data and historical data. Notice, however, the HyperTrend object class is not available in Lookout Express systems. See *HyperTrend* object class definition for more information.

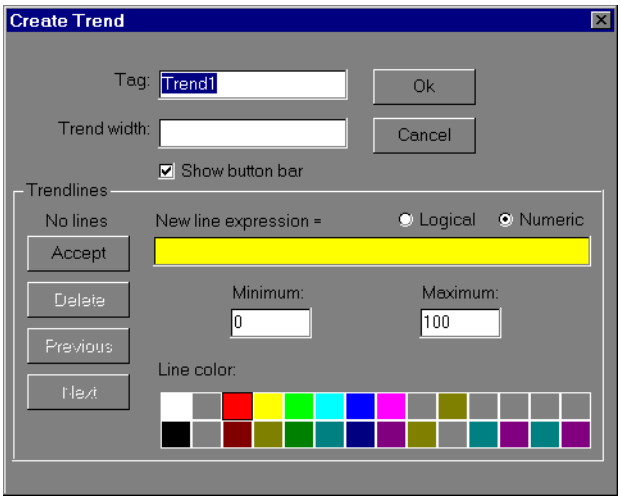


Figure 18-129. Trend definition parameters dialog box

Trend width is the width of the graph in units of time. Graphs may have a width, or time span, of anywhere from two seconds to one year. The **Trend width** in the example dialog box indicates a time span of 1:00:00 or one hour. See *Numeric Data Members* in Chapter 5, *Developer Tour*, for more information on entering time constants.

Sample interval specifies the frequency at which values are captured for plotting. The Trend object divides **Sample interval** into **Trend width** to determine how many data points to temporarily save for each trend line. Lookout stores these data points within the trend object, in RAM. For this reason, you should carefully choose **Sample interval**. Typically, no more

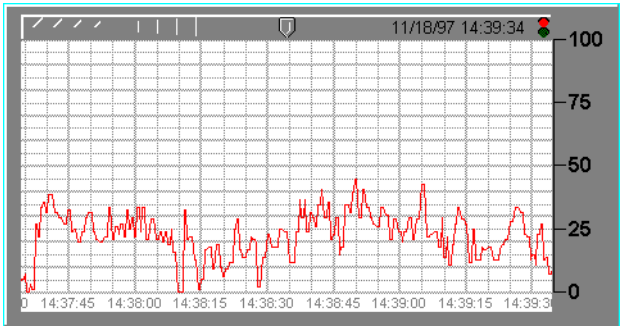
than 100 to 200 points need to be stored for each trend line, less for data that does not change rapidly. Because most computer display resolutions cannot take advantage of more than about 800 data points, it is rarely necessary to exceed that limit.

The **Trendlines** parameters are used to add, modify, or delete expressions from the trend graph. The data field to the right of the Accept button is used to enter logical and numeric expressions for plotting. The **Logical** and **Numeric** selections must be set to correspond with the current expression result.

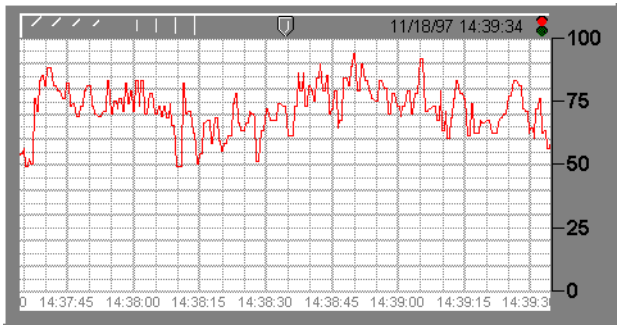
Line color specifies the color of the trend line for the current expression.

Minimum and **Maximum** settings determine where on the trend graph the expression is plotted. **Minimum** is the bottom of the graph while **Maximum** is the top of the graph—regardless of the range of the expression. These settings create an imaginary vertical scale and affect each expression independently, so that you can plot trend lines on any section of a trend graph.

For example, take two numeric expressions, both of which range from 0 to 50. Set the **Minimum** and **Maximum** to 0 and 100 on the first expression, and –50 and 50 on the second. The first expression is plotted in the bottom half of the chart while the second expression is plotted in the top half of the chart, even though they both fluctuate between 0 and 50.

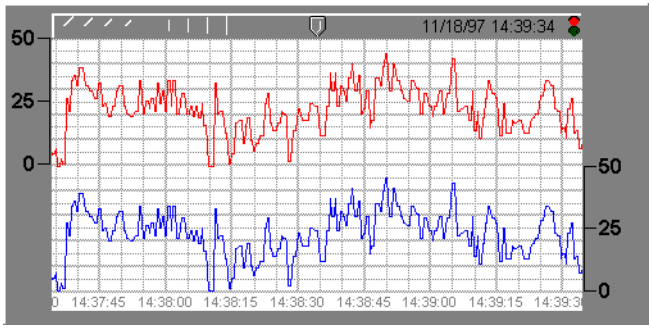


This shows the imaginary scale of the first expression (where min.=0 and max.=100). Because the expression ranges from 0 to 50, it is plotted in the bottom half of the graph.



This figure shows the imaginary scale of the second expression (where min.=–50 and max.=50). Because the expression ranges from 0 to 50, it is plotted in the top half of the graph.

When both expressions are entered on a single trend graph, you get the following effect. Notice the custom scales on either end of the graph.



If **Logical** is selected for the expression type, the minimum and maximum settings change to **Position** and **Height**. These two values now represent a number between 0% and 100%, and determine the baseline location of the trend line and its unit height when the expression goes TRUE.

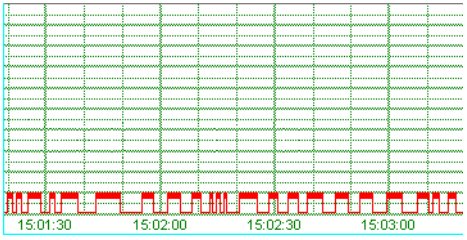


Figure 18-130. Plot of a Logical Value

When you finish entering or modifying the trend line parameters, select the **Accept** button. This adds the expression to the **Trendlines** list. The **Delete** button is used to delete the current expression from the trend graph. The **Previous** and **Next** buttons toggle you through a list of all the expressions named for the current trend object.

You can display trend graphs in various colors with different timeline styles and grid spacing.

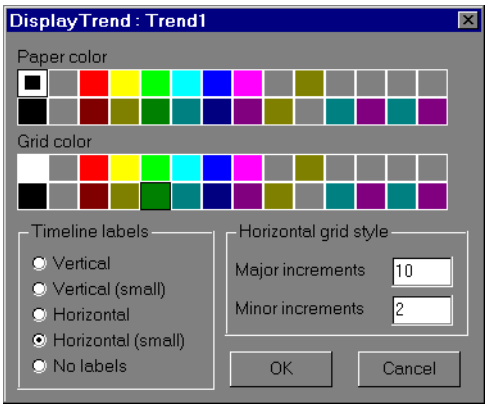


Figure 18-131. Trend Display Parameters Dialog Box

Timeline labels determine where and how the date and time are to be displayed on the trend graph.

Major increments specifies the number of heavy horizontal grid lines on a trend graph. This value is independent of the range of any trend expressions.

Minor increments specifies the number of light horizontal grid lines between the major increment grid lines on a trend graph. This value is independent of the range of any trend expressions.

Trend Data Members

Table 18-133. Trend Data Members

Data Members	Type	Read	Write	Description
visible	logical	no	yes	When TRUE, the trend becomes visible on the control panel. When FALSE, the trend is invisible. The default value is TRUE.

Comments ***Any number of trend lines can be displayed on a given trend—however too many lines can confuse the information you are trying to display. Consider this when deciding how many lines to show on one Trend.***

Trends should usually be displayed with custom scales along the vertical axes.

Trend displays are updated as quickly as once per second, depending on screen resolution, the size of the trend graph, and the trend width setting. Computers with slow display adapters may be slowed down considerably when a large trend graph is displayed that is being updated once per second. On slower computers with slow display cards (no graphics coprocessor), consider limiting the size of fast-moving Trends to less than one fourth the screen size.

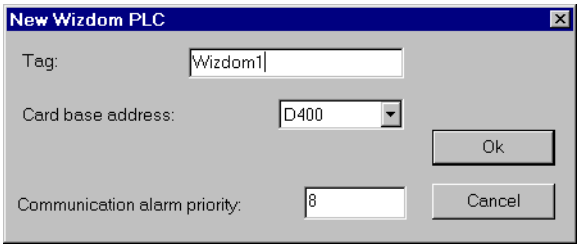
Trend objects save trend line data points to the Lookout state file (`.LST`) periodically as defined in the System Options dialog box, and any time you close your process file or exit Lookout. For this reason, you do not lose the plot lines shown on your trends if you exit Lookout or if your computer goes down.

Wizdom

Wizdom is an object class that Lookout uses to communicate with the Wizdom Coprocessor Card. Create one Wizdom object for each Wizdom card installed in the computer.

The Wizdom object scans configured data members for changes 10 times per second.

Wizdom cards are single-board computers. They communicate with the PC through dual-ported memory (that is, RAM accessed by both the PC processor and the Wizdom card processor).



In this example, Jumper settings on the card are configured to use hardware interrupt 7 and beginning memory address D000.

Card base address specifies the beginning memory location of the dual ported RAM address. It should match jumper settings on the card. The card uses 8K of physical memory.

When the Wizdom card writes an input value into a memory location, the card generates an interrupt (if so configured), causing the Lookout Wizdom object to immediately scan. This ensures faster responses to field changes.

☞ **Note** ***Be sure to verify that no other drivers are mapped to the selected memory location and interrupt.***

☞ **Note** ***Because the EMM386 memory manager only recognizes the first 4K of mapped memory, it is important to add a memory exclude statement to your computer*** `CONFIG.SYS` ***file as instructed in the card documentation.***

Wizdom Data Members

Like other object classes designed to communicate with external I/O, Wizdom objects can contain a great deal of data. All readable and writable members (inputs/outputs), polling instructions, and so on are bundled with

the object. As soon as you create a Wizdom object you immediately have access to all the object data members.

Table 18-134. Wizdom Data Members

Data Members	Type	Read	Write	Description
B0 – B1903	numeric	yes	yes	8-Bit word ranging from 0 to 255
B0.0 – B1903.7	logical	yes	yes	1 Bit in an 8-Bit word
BCD0.2 – BCD1900.16				Reserved for future binary coded decimal implementation
BS0 – BS1903	numeric	yes	yes	Signed 8-bit word ranging from –128 to 127
BS0.0 – BS1903.7	logical	yes	yes	1 Bit in a signed 8-bit word
DF0 – DF1896	numeric	yes	yes	64-bit IEEE floating point double precision word
DS0 – DS1900	numeric	yes	yes	Signed 32-bit double-precision word ranging from –2,147,483,648 to 2,147,483,648
DS0.0 – DS1900.31	logical	yes	yes	1 Bit in a signed 32-bit word
DW0 – DW1900	numeric	yes	yes	32-Bit double-precision word ranging from 0 to 4,294,967,295
DW0.0 – DW1900.31	logical	yes	yes	1 Bit in a 32-bit word
F0 – F1900	numeric	yes	yes	32-bit IEEE floating point word
S0 – S1902	numeric	yes	yes	Signed 16-bit word ranging from –32,768 to 32,767
S0.0 – S2047.15	logical	yes	yes	1 Bit in a signed 16-bit word
STR0 – STR1900				Reserved for future character string implementation
W0 – W1902	numeric	yes	yes	16-Bit word ranging from 0 to 65,535
W0.0 – W1902.15	logical	yes	yes	1 Bit in a 16-Bit word

Comments ***The data member addresses listed in the data member table all share the same 4K of memory. For this reason, the following are all the same bit:***

B391.5 = W391.5 = DS391.5

XBarR

The XBarR object class is one of the Lookout Statistical Process Control (SPC) tools and can play an important role in your Total Quality Management (TQM) program. This object class graphically displays an *X-Bar chart* and/or an *R chart* for a given signal. These control charts (also called mean and range charts, respectively) enable you to track your process to determine if it is about to go out of control and needs corrective attention.

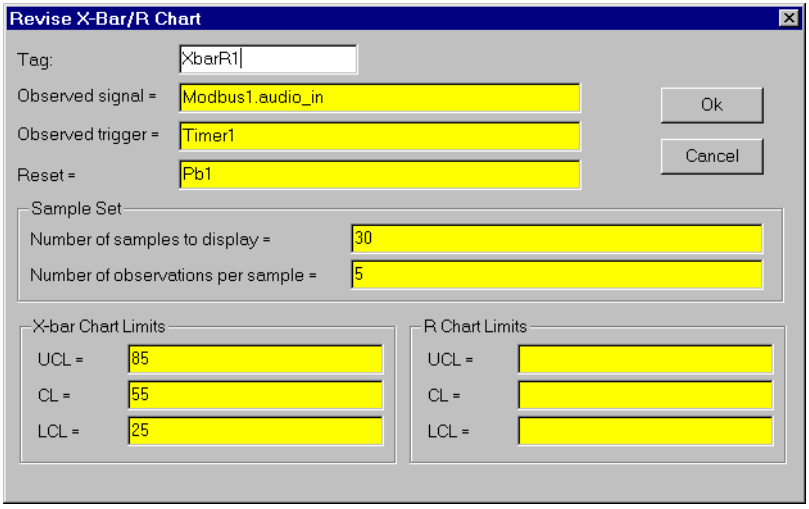


Figure 18-132. XBarR Definition Parameters Dialog Box

The XBarR object reads the **Observed signal** value any time the **Observed trigger** transitions from OFF to ON. When the object collects the specified **Number of observations per sample** (2 – 25), it calculates the mean and range and plots them on their respective charts. Each chart plots the specified **Number of samples to display**.

X-bar Chart limits and **R Chart limits** can be specified or calculated, as you choose. The object continually calculates chart limits based on new samples as they are accumulated. These calculated limits enable you to track your process for a time to determine limit settings that your process is capable of operating within. Once you know what limits your process can normally handle, you can specify the chart limit parameters by entering them directly.

LCL and **UCL** chart limits identify the lower and upper control limits. These outer limits generally define the range of process acceptability. **CL** identifies the *centerline*. For the X-bar chart, the centerline is the mean of sample means. For the R chart, the centerline is the average of the sample ranges.

You can display X-bar charts and R charts in various colors with different label styles and grid spacing.

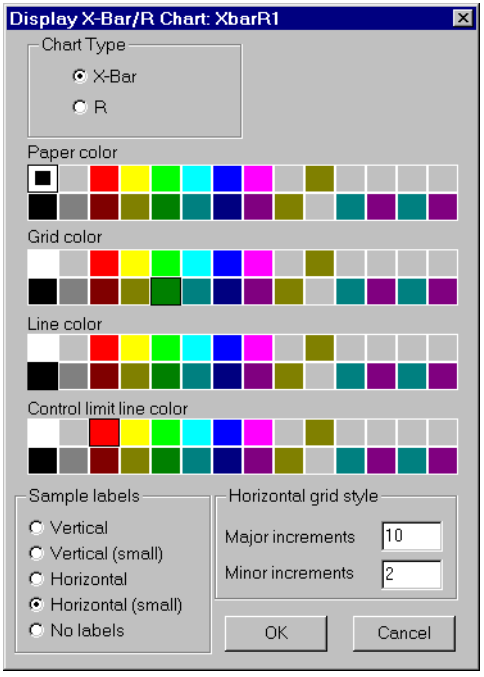


Figure 18-133. XBarR Display Parameters Dialog Box

When you enter the **LCL**, **UCL** and **CL** chart limit parameters for a chart, the entered values represent the limits shown on the chart. The following X-Bar Chart shows upper control limit, center line, and lower control limit entered as constants. The chart shows that the majority of the last 30 samples are running a bit under the centerline, suggesting that the process may need to be adjusted. The calculated control limits, shown below the graph, indicate new limits that might be used based on the current plotted samples.

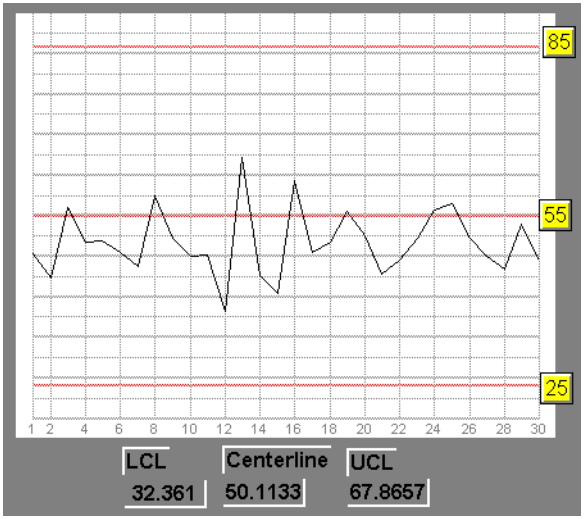


Figure 18-134. X-Bar Chart Showing Upper Control Limit, Center Line, and Lower Control Limits.

If you leave one or more of the limit parameters blank for a chart in the definition parameters dialog box, the chart shows its *calculated* limits. As you can see in the R chart below, the calculated limits and the plotted limits are the same.

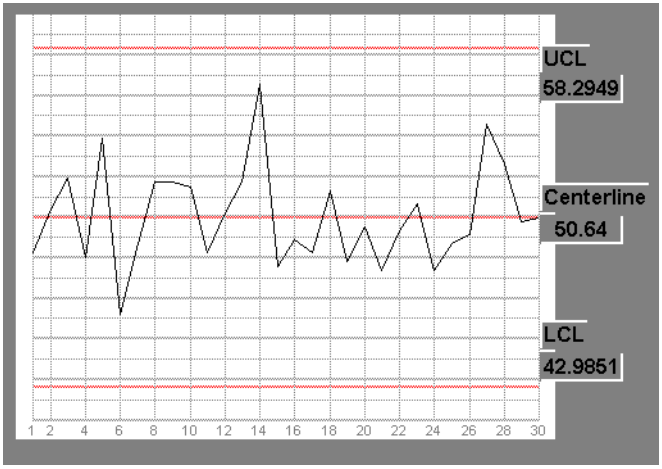


Figure 18-135. R Chart Showing Upper Control Limit, Center Line, and Lower Control Limits as Calculated Based on Plotted Samples.

XBarR Data Members

Table 18-135. XBarR Data Members

Data Members	Type	Read	Write	Description
R_ChartMax	numeric	yes	no	Identifies the top of the Range chart graph (the value of the plotted line when it is at 100 percent of the Y axis). The object recalculates this value as each new sample is read to ensure that the plotted line is always visible on the graph.
R_ChartMin	numeric	yes	no	Identifies the bottom of the Range chart graph (the value of the plotted line when it is at zero percent of the Y axis). The object recalculates this value as each new sample is read to ensure that the plotted line is always visible on the graph.
R_CL	numeric	yes	no	Centerline of range chart showing average of sample ranges calculated as follows: $$\overline{R} = \frac{\sum_{i=1}^{n} R_i}{n}$$
R_LCL	numeric	yes	no	Lower control limit of Range chart, calculated using the sampled data as follows: $$LCL = D_3\overline{R}$$ where D_3 is from any standard 3-sigma control factors table
R_Sample	numeric	yes	no	Calculated range, R, of the last completed sample of observations, defined as the difference between the maximum and minimum observations in the sample.

Table 18-135. XBarR Data Members (Continued)

Data Members	Type	Read	Write	Description
R_UCL	numeric	yes	no	Upper control limit of Range chart, calculated using the sampled data as follows: $UCL = D_4\overline{R}$ where D_4 is from any standard 3-sigma control factors table
Visible	logical	yes	yes	When TRUE, the X-bar and R charts become visible on the control panel. When FALSE, they are invisible. The default value is TRUE.
Xbar_ChartMax	numeric	yes	no	Identifies the top of the X-bar chart graph (the value of the plotted line if it is at 100 percent of the Y axis). The object recalculates this value as each new sample is read to ensure that the plotted line is always visible on the graph.
Xbar_ChartMin	numeric	yes	no	Identifies the bottom of the X-bar chart graph (the value of the plotted line if it is at zero percent of the Y axis). The object recalculates this value as each new sample is read to ensure that the plotted line is always visible on the graph.
Xbar_CL	numeric	yes	no	Centerline of X-bar chart showing the mean of sample means, calculated as follows: $\overline{\overline{x}} = \frac{\sum_{i=1}^{n} \overline{x}_i}{n}$

Table 18-135. XBarR Data Members (Continued)

Data Members	Type	Read	Write	Description
Xbar_LCL	numeric	yes	no	Lower control limit of X-bar chart, calculated as follows: $LCL = \overline{\overline{x}} - A_2 \overline{R}$ where A_2 is from any standard 3-sigma control factors table
Xbar_Sample	numeric	yes	no	Calculated mean of the last completed sample of observations, where $\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n}$
Xbar_UCL	numeric	yes	no	Upper control limit of X-bar chart, calculated as follows: $UCL = \overline{\overline{x}} + A_2 \overline{R}$ where A_2 is from any standard 3-sigma control factors table

Related Objects ***Average, Histogram, Maximum, Minimum, Sample***

Related Functions ***Avg, Max, Min, Stdev, Stdevp, Sum, Var, Varp***

XChart

The XChart object class graphically plots the value of an observed signal on a chart in response to a trigger signal.

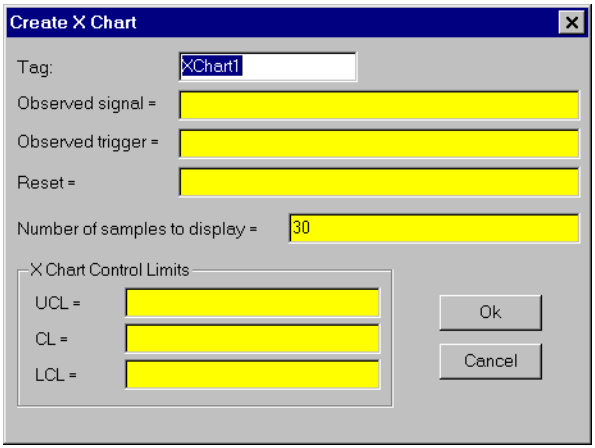


Figure 18-136. XChart Definition Parameters Dialog Box

The XChart object reads the value of the **Observed signal** any time the **Observed Trigger** transitions from `OFF` to `ON`.

Xchart plots the specified **Number of samples to display**. When it reaches the limit set by that parameter, it removes the oldest point before plotting the newest one. **Number of samples to display** must be between 2-4000. All the points on the chart are cleared any time **Reset** transitions from `OFF` to `ON`.

XChart Control Limits can be specified or calculated as you choose. If you leave the values in this section of the dialog box open, the object continually calculates chart limits based on new values as they are accumulated. You can watch your process for a time with XChart calculating limits to determine the ranges of normal operation, and then you can set the chart limits directly. Once set, XChart displays limit lines at the specified settings.

UCL and **LCL** identify the upper and lower control limit lines. **CL** defines the centerline.

You can display Xcharts in various colors and styles with different vertical and horizontal grid spacing.

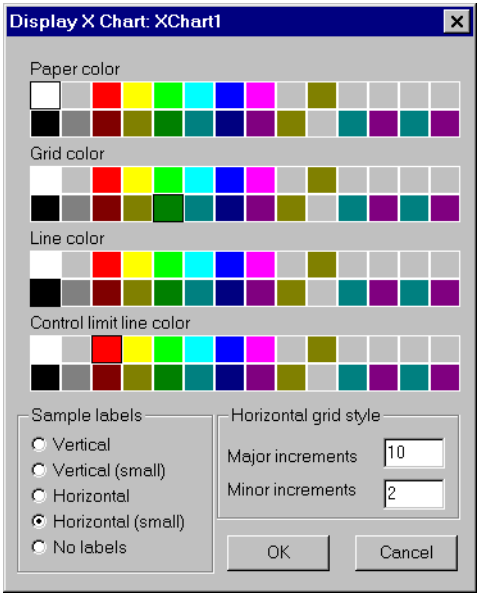


Figure 18-137. XChart Display Parameters Dialog Box

XChart Data Members

Table 18-136. Wizdom Data Members

Data Members	Type	Read	Write	Description
ChartMax	numeric	yes	no	identifies the top of the chart graph (the value of the plotted line when it is at 100 percent of the Y axis)
ChartMin	numeric	yes	no	identifies the bottom of the chart graph (the value of the plotted line when it is at zero percent of the Y axis)

Table 18-136. Wizdom Data Members (Continued)

Data Members	Type	Read	Write	Description
UCL	numeric	yes	no	upper control limit of chart. When calculated, is drawn 3 standard deviations above CL, using the following formula: $UCL = CL + 3\sqrt{\frac{\sum_{i=1}^{n}(x_i)^2 - \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n}}{n-1}}$
CL	numeric	yes	no	centerline of the chart, showing the mean of the sample, calculated by the formula: $CL = \frac{\sum_{i=1}^{n} x_i}{n}$
LCL	numeric	yes	no	lower control limit of the chart (3 standard deviations below CL), calculated by the formula: $LCL = CL - 3\sqrt{\frac{\sum_{i=1}^{n}(x_i)^2 - \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n}}{n-1}}$
Sample	numeric	yes	no	value of the last sample recorded
Visible	logical	yes	yes	When TRUE, XChart become visible on the control panel. When FALSE, XChart is invisible. The default value is TRUE.

XYChart

The XYChart object class graphically plots X andY points on a chart in response to a trigger signal.

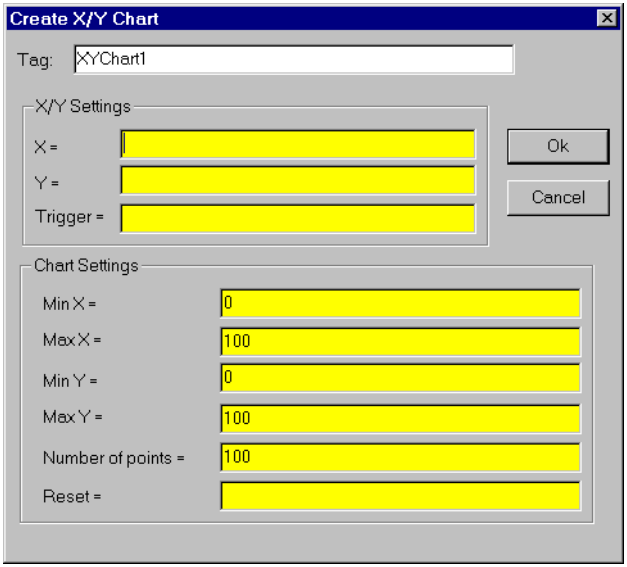


Figure 18-138. XYChart Definition Parameters Dialog Box

The XYChart object reads the value of the **X** and **Y** signals any time **Trigger** transitions from `OFF` to `ON` and plots the corresponding X/Y point on the chart in relation to the chart ranges **Min X**, **Max X**, **Min Y**, and **Max Y**.

XYChart plots the specified **Number of points**. When it reaches that limit, it removes the oldest point before plotting the newest one. **Number of points** must be between 2-4000. All the points on the chart are cleared any time **Reset** transitions from `OFF` to `ON`.

You can display X/Y charts in various colors and styles with different vertical and horizontal grid spacing.

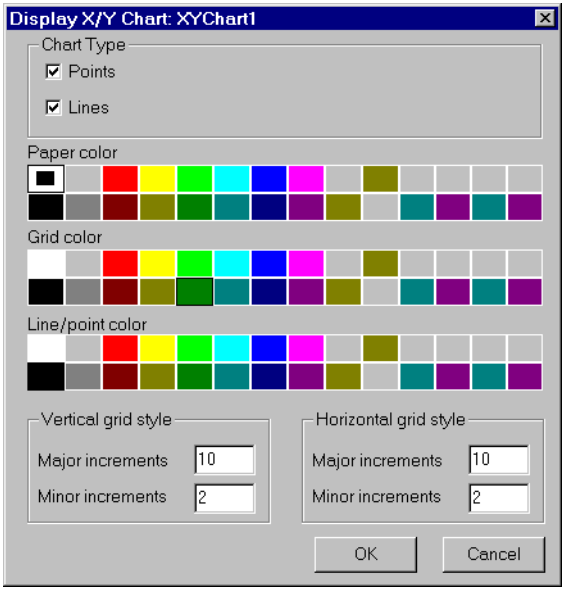


Figure 18-139. XYChart Display Parameters Dialog Box

XYChart Data Members

Table 18-137. XYChart Data Members

Data Members	Type	Read	Write	Description
ChartXMax	numeric	yes	no	Identifies the right most point of the chart graph (the value of the plotted line when it is at 100 percent of the X axis).
ChartXMin	numeric	yes	no	Identifies the left most point of the chart graph (the value of the plotted line when it is at zero percent of the X axis).
ChartYMax	numeric	yes	no	Identifies the top of the chart graph (the value of the plotted line when it is at 100 percent of the Y axis).

Table 18-137. XYChart Data Members (Continued)

Data Members	Type	Read	Write	Description
ChartYMin	numeric	yes	no	Identifies the bottom of the chart graph (the value of the plotted line when it is at zero percent of the Y axis).
Visible	logical	yes	yes	When TRUE, X/Y chart become visible on the control panel. When FALSE, it is invisible. The default value is TRUE.

A

Customer Communication

For your convenience, this appendix contains forms to help you gather the information necessary to help us solve your technical problems and a form you can use to comment on the product documentation. When you contact us, we need the information on the Technical Support Form and the configuration form about your system configuration to answer your questions as quickly as possible.

National Instruments has technical assistance through electronic, fax, and telephone systems to quickly provide the information you need. Our electronic services include a bulletin board service, an FTP site, a fax-on-demand system, and e-mail support. If you have a hardware or software problem, first try the electronic support systems. If the information available on these systems does not answer your questions, we offer fax and telephone support through our technical support centers, which are staffed by applications engineers.

Electronic Services

Bulletin Board Support

National Instruments has BBS and FTP sites dedicated for 24-hour support with a collection of files and documents to answer most common customer questions. From these sites, you can also download the latest instrument drivers, updates, and example programs. For recorded instructions on how to use the bulletin board and FTP services and for BBS automated information, call 512 795 6990. You can access these services at:

United States: 512 794 5422
Up to 14,400 baud, 8 data bits, 1 stop bit, no parity

United Kingdom: 01635 551422
Up to 9,600 baud, 8 data bits, 1 stop bit, no parity

France: 01 48 65 15 59
Up to 9,600 baud, 8 data bits, 1 stop bit, no parity

FTP Support

To access our FTP site, log on to our Internet host, `ftp.natinst.com`, as `anonymous` and use your Internet address, such as `joesmith@anywhere.com`, as your password. The support files and documents are located in the `/support` directories.

Fax-on-Demand Support

Fax-on-Demand is a 24-hour information retrieval system containing a library of documents on a wide range of technical information. You can access Fax-on-Demand from a touch-tone telephone at 512 418 1111.

E-Mail Support (Currently USA Only)

You can submit technical support questions to the applications engineering team through e-mail at the Internet address listed below. Remember to include your name, address, and phone number so we can contact you with solutions and suggestions.

`support@natinst.com`

Telephone and Fax Support

National Instruments has branch offices all over the world. Use the list below to find the technical support number for your country. If there is no National Instruments office in your country, contact the source from which you purchased your software to obtain support.

Country	Telephone	Fax
Australia	03 9879 5166	03 9879 6277
Austria	0662 45 79 90 0	0662 45 79 90 19
Belgium	02 757 00 20	02 757 03 11
Brazil	011 288 3336	011 288 8528
Canada (Ontario)	905 785 0085	905 785 0086
Canada (Quebec)	514 694 8521	514 694 4399
Denmark	45 76 26 00	45 76 26 02
Finland	09 725 725 11	09 725 725 55
France	01 48 14 24 24	01 48 14 24 14
Germany	089 741 31 30	089 714 60 35
Hong Kong	2645 3186	2686 8505
Israel	03 6120092	03 6120095
Italy	02 413091	02 41309215
Japan	03 5472 2970	03 5472 2977
Korea	02 596 7456	02 596 7455
Mexico	5 520 2635	5 520 3282
Netherlands	0348 433466	0348 430673
Norway	32 84 84 00	32 84 86 00
Singapore	2265886	2265887
Spain	91 640 0085	91 640 0533
Sweden	08 730 49 70	08 730 43 70
Switzerland	056 200 51 51	056 200 51 55
Taiwan	02 377 1200	02 737 4644
United Kingdom	01635 523545	01635 523154
United States	512 795 8248	512 794 5678

Technical Support Form

Photocopy this form and update it each time you make changes to your software or hardware, and use the completed copy of this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

If you are using any National Instruments hardware or software products related to this problem, include the configuration forms from their user manuals. Include additional pages if necessary.

Name _____

Company _____

Address _____

_____

Fax (___) _____Phone (___) _____

Computer brand_____ Model _____Processor_____

Operating system (include version number) _____

Clock speed _____MHz RAM _____MB Display adapter _____

Mouse ___yes ___no Other adapters installed_____

Hard disk capacity _____MB Brand_____

Instruments used _____

_____

National Instruments hardware product model_____ Revision _____

Configuration _____

National Instruments software product_____ Version _____

Configuration _____

The problem is: _____

_____

_____

_____

_____

List any error messages: _____

_____

_____

The following steps reproduce the problem: _____

_____

_____

_____

_____

Lookout Hardware and Software Configuration Form

Record the settings and revisions of your hardware and software on the line to the right of each item. Complete a new copy of this form each time you revise your software or hardware configuration, and use this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

National Instruments Products

Lookout Version _____

Other National Instruments software and version _____

Hardware name and revision _____

Interrupt level of hardware _____

DMA channels of hardware _____

Base I/O address of hardware _____

Other devices in system _____

Base I/O address of other devices _____

DMA channels of other devices _____

Interrupt level of other devices _____

Other Products

Computer make and model _____

Microprocessor _____

Clock frequency or speed _____

Type of video board installed _____

Operating system version _____

Operating system mode _____

Other applications running on the system _____

Programming language _____

Programming language version _____

Network hardware _____

Networking protocols _____

PLCs in use _____

RTUs in use _____

Other boards in system _____

Base I/O address of other boards _____

DMA channels of other boards _____

Interrupt level of other boards _____

Other _____

Documentation Comment Form

National Instruments encourages you to comment on the documentation supplied with our products. This information helps us provide quality products to meet your needs.

Title: *Lookout™ Reference Manual*

Edition Date: March 1998

Part Number: 321254B-01

Please comment on the completeness, clarity, and organization of the manual.

_____

_____

_____

_____

_____

_____

_____

If you find errors in the manual, please record the page numbers and describe the errors.

_____

_____

_____

_____

_____

_____

_____

Thank you for your help.

Name _____

Title _____

Company _____

Address _____

_____

E-Mail Address _____

Phone (___) _____ Fax (___) _____

Mail to: Technical Publications
National Instruments Corporation
6504 Bridge Point Parkway
Austin, Texas 78730-5039

Fax to: Technical Publications
National Instruments Corporation
512 794 5678

Index

A

AB object classes (AB_PLC2, AB_PLC5, AB_SLC500), 18-2 to 18-21
  Allen-Bradley Parameters dialog box, 18-3 to 18-4
  communication with Allen-Bradley controllers
    PLC-2 family, 18-2
    PLC-5 family, 18-2
    SLC-500 family, 18-2 to 18-3
  data members, 18-9 to 18-16
    AB_PLC2 (table), 18-10
    AB_PLC5 (table), 18-13 to 18-16
    AB_SLC500 (table), 18-10 to 18-13
  DH+ interface parameters, 18-5 to 18-8
  error messages, 18-16 to 18-21
  Ethernet interface parameters, 18-8
  serial port interface parameters, 18-4
  using 5136-SD card from S-S Technologies, Inc., 18-9
ABS function, 7-17
absolute dates and times (table), 2-10
accounts, 10-1 to 10-3
  assigning security levels (note), 10-2
  definition, 10-1
  forgetting your password (note), 10-3
  modifying, 10-2 to 10-3
Accounts command, Options menu, 16-8 to 16-9
Accumulator object class, 18-22 to 18-23
  data member (table), 18-23
  Definition Parameters dialog box, 18-22
  description, 18-22
Acknowledge command, Alarms menu, 16-14
acknowledging alarms, 9-11 to 9-12
ACOS function, 7-25
action verification, 10-8
active notification, 2-14 to 2-15
Add command, Run menu, 16-10 to 16-11
AdvantechPCL object class, 18-24 to 18-26
  data members (table), 18-25 to 18-26
  Definition Parameters dialog box, 18-24 to 18-25
Alarm commands, 16-11 to 16-14
  Acknowledge, 16-14
  Deselect All, 16-14
  Display Options, 16-12
  Filter Options, 16-12 to 16-13
  Print, 16-13
  Select All, 16-14
  Show, 16-12
$Alarm global object, 18-30 to 18-32
  data members (table), 18-31 to 18-32
  description, 18-30 to 18-31
  Edit Connections dialog box, 18-30
  using with other objects, 18-32
Alarm object class, 18-27 to 18-29
  creating alarm objects, 9-1 to 9-2
  data members (table), 18-29
  typical settings
    logical style alarm, 18-27 to 18-28
    numeric style alarm, 18-28
alarms, 9-1 to 9-12
  acknowledging, 9-11 to 9-12
  alarm subsystem, 9-3 to 9-12
  circular reference alarms, 9-2
  database-generated alarms, 9-1 to 9-2
  DDE, 13-6 to 13-7
  defining alarm conditions, 9-1 to 9-3
  deselecting, 9-11
  display options, 9-7
  filters, 9-8 to 9-9
  groups, 9-3
  National Instruments Fieldbus, 18-223
  overview, 2-16

Philips object class, 18-285
printing, 16-4
priorities, 9-4
S5_3964 object class, 18-340 to 18-341
selecting, 9-11
setting logging option, 16-7
Alarms window, 9-5 to 9-6
color scheme (table), 9-6
illustration, 9-5
viewing alarms, 9-5
aliases
OPC Client Driver
adding single alias, 18-236 to 18-237
importing alias lists, 18-238 to 18-239
optional use of (note), 4-10
purpose and use, 4-9 to 4-10
Align command, Arrange menu, 17-17
Allen-Bradley PLC controllers. *See* AB object classes.
Alternator object class, 18-33 to 18-38
Command and Advance data members, 18-35 to 18-36
connecting, 18-35
data members (table), 18-34
Definition Parameters dialog box, 18-33 to 18-34
Delay between device starts parameter, 18-36
description, 18-33 to 18-34
ElapsedTime data member, 18-37
Hand - Off - Auto modes, 18-36 to 18-37
Maximum run time parameter, 18-36
RunTime data member, 18-37
status messages, 18-37 to 18-38
AND function, 7-14
Animator object class, 18-39 to 18-43
color animations, 18-41 to 18-43
creating moving animations, 18-40 to 18-41
data members, 18-43
displaying dynamic graphics (table), 8-6
Select graphic dialog box, 18-39
Applicom object classes, 18-44 to 18-66
corresponding protocols/devices (table), 18-45 to 18-46
data members, 18-46 to 18-61
April 1000 (table), 18-47 to 18-48
JBUS (table), 18-46 to 18-47
Klockner-Moeller (table), 18-48 to 18-49
local (table), 18-46
Otic Fischer & Porter (table), 18-49
Profibus DP (table), 18-50
Profibus FMS (table), 18-51
Profibus L2 (table), 18-52 to 18-53
SAIA SBus (table), 18-54
Siemens H1 (table), 18-59 to 18-60
Siemens S5 AS511 (table), 18-55 to 18-56
Siemens S7 MPI (table), 18-56 to 18-58
Siemens S7 PPI (table), 18-58 to 18-59
Telemecanique (table), 18-61
Definition Parameters dialog box, 18-44 to 18-45
general information on using drivers, 18-61 to 18-64
configuration of Applicom server, 18-63
creating cyclic functions, 18-64
loading Applicom server, 18-63
local and image modes, 18-62
testing Applicom server, 18-63 to 18-64
special instructions for local object class, 18-64 to 18-65
status messages, 18-65 to 18-66
APT tag file, importing, 16-10, 18-402

Aquatrol object class, 18-67 to 18-71
data members, 18-69 to 18-70
Definition Parameters dialog box, 18-68 to 18-69
RTU Configuration dialog box, 18-68 to 18-69
status messages, 18-70 to 18-71
architecture of Lookout, 2-2 to 2-17
arithmetic operators, 7-10
Arrange commands, 17-16 to 17-18
Align, 17-17
Group, 17-18
Move to Back, 17-18
Move to Front, 17-18
Space Evenly, 17-17 to 17-18
Ungroup, 17-18
Arrange Icons command, Windows menu, 16-14
ASCII object class, 18-72 to 18-81. *See also* IPASCII object class.
data members (table), 18-73 to 18-74
Definition Parameters dialog box, 18-72 to 18-73
error messages, 18-80 to 18-81
request and response format strings, 18-75 to 18-79
entering format string, 18-78
markers, 18-76 to 18-78
request frame construction examples, 18-79
response format examples, 18-79
using sum data members, 18-80
ASIN function, 7-25
ATAN function, 7-25
ATAN2 function, 7-25
automatic process loading, 1-5 to 1-6
Average object class, 18-82 to 18-83
data members, 18-82
Definition Parameters dialog box, 18-82
AVG function, 7-20

B

background color, changing, 17-20
bitmap (.BMP) graphics
compared with Windows metafiles, 8-17 to 8-18
displaying, 8-3 to 8-5
.BMP files. *See* bitmap (.BMP) graphics.

C

Change functions, 17-19 to 17-21
Background Color, 17-20
Font, 17-19
Justify Text, 17-21
Numeric Format, 17-20
Text Color, 17-19 to 17-20
Change Registration Info button, 16-8
circular reference alarms, 9-2
Citadel Historical Database Logger, 11-5 to 11-9
accessing data with ODBC driver, 12-1 to 12-12
configuring ODBC driver, 12-1 to 12-3
data transforms, 12-4 to 12-5
SQL examples, 12-5 to 12-6
traces table, 12-3 to 12-4
using Microsoft Access, 12-10 to 12-11
using Microsoft Excel, 12-9 to 12-10
using Microsoft Query, 12-6 to 12-9
using Microsoft Visual Basic, 12-12
creating historical database, 11-6 to 11-8
data location, 11-6
information overload, 11-8 to 11-9
logging criteria, 11-8
client, DDE, 13-3 to 13-4
Close command, File menu, 16-2
color grids in dialog boxes (note), 5-6

colors
alarm status (table), 9-6
animations, 18-41 to 18-43
changing
background color, 17-20
text color, 17-19 to 17-20
using in graphics, 8-3
commands. *See* Edit mode menu commands; Runtime menu commands.
comma-separated value (.CSV) files
for Spreadsheet Logger, 11-3 to 11-4
SqlExec object class, 18-370
communications service. *See* serial communications.
comparison operators, 7-11 to 7-13
configuration shortcuts, 3-8 to 3-10
mouse, 3-8 to 3-9
remembering tagnames, 3-9 to 3-10
connecting objects, 4-19 to 4-23
Alternator object class, 18-35
data members to data members, 4-21 to 4-23
data members to parameters, 4-19 to 4-20
example (figure), 2-12
identifying object data members, 4-20 to 4-21
overview, 2-12
PLC and RTU (development process example), 5-20 to 5-23
selecting target object (note), 4-19
signals, DataTable object class, 18-94 to 18-96
supervisory control, 2-12
control objects, adding to process file (example), 5-11 to 5-16
Control Panel command, Insert menu, 17-8 to 17-9
control panels, 3-4 to 3-5. *See also* Panel object class.
DataTables example
display panel, 18-96 to 18-97
multiplexing displays and graphics, 18-91 to 18-92
operating multiplexed panel, 18-97 to 18-98
development example
completing, 5-17 to 5-20
creating, 5-1 to 5-2
displaying
data members, 4-24 to 4-25
expressions, 7-3 to 7-4
inserting, 17-8 to 17-9
Normal, 3-4
overview, 3-4
Popup, 3-4
positioning inserted items, 17-9 to 17-10
report generation, 11-10 to 11-12
screen resolution for display panels (note), 5-2
viewing security, 10-5
controllable objects
networking considerations, 14-3 to 14-5
security considerations, 10-4 to 10-5
viewing security, 10-6
Copy command, Edit menu, 17-2
copying and pasting objects (note), 4-4
copying object databases, 4-19
COS function, 7-26
Counter object class, 18-82
data members (table), 18-82
Definition Parameters dialog box, 18-82
Create Alarm dialog box, 9-2 to 9-3
Create command, Object menu, 17-11 to 17-13
creating objects, 4-2 to 4-5
.CSV (comma-separated values) files, 11-3 to 11-4, 18-370
CTS timeout setting, 6-5
cursors, for DataTables, 18-98 to 18-99
multiple, 18-99 to 18-100
custom graphics. *See also* graphics.
creating, 8-13 to 8-16

step-by-step example, 8-13 to 8-16
displaying static graphics, 8-3 to 8-5
exporting to Lookout, 8-15
testing in Lookout, 8-15 to 8-16
customer communication, A-1 to A-2
Cut command, Edit menu, 17-2
Cutler-Hammer object class, 18-85 to 18-89
data members (table), 18-87 to 18-88
Definition Parameters dialog box, 18-85 to 18-87
status messages, 18-88 to 18-89

D

data display, adding to process file (example), 5-9 to 5-10
Data files location option, System command, 16-8
data logging
Citadel Historical Database Logger, 11-5 to 11-9
report generation, 11-10 to 11-12
Spreadsheet Logger, 11-1 to 11-5
data members
AB object classes (table), 18-9 to 18-16
AdvantechPCL object class (table), 18-25 to 18-26
$Alarm global object (table), 18-31 to 18-32
Alarm object class (table), 18-29
Alternator object class (table), 18-34
Animator object class (table), 18-43
Applicom object classes, 18-46 to 18-61 (table)
Aquatrol object class (table), 18-69 to 18-70
ASCII object class (table), 18-73 to 18-74
Average object class (table), 18-82
connecting
identifying data members, 4-20 to 4-21
to data members, 4-21 to 4-23
to parameters, 4-19 to 4-20
Counter object class (table), 18-82
Cutler-Hammer object class (table), 18-87 to 18-88
DataTable object class (table), 18-100 to 18-102
DdeLink object class (table), 18-104
DdeTable object class (table), 18-108
definition, 2-4, 2-5
DelayOff object class (table), 18-110
DelayOn object class (table), 18-112
DeltaTau object class (table), 18-113 to 18-114
Derivative object class (table), 18-116
DialGauge object class (table), 18-119
displaying on control panels, 4-24 to 4-25
DL205 and DL405 object classes (table), 18-123 to 18-125
Dynamic object class (table), 18-129 to 18-132
ElapsedTime object class (table), 18-133
error messages, 18-230 to 18-232
Event object class (table), 18-134
Expression object class (table), 18-136
FisherROC object class (table), 18-139 to 18-141
Flipflop object class (table), 18-144
Gauge object class (table), 18-146
GE_Series6 object class (table), 18-149 to 18-150
GE_Series90 object class (table), 18-153 to 18-155
Histogram object class (table), 18-159 to 18-160
Hitachi object class (table), 18-162 to 18-164
HyperTrend object class (table), 18-171 to 18-172
(implicit) data members, 2-11
Integral object class, 18-174

Interpolate object class (table), 18-178
Interval object class (table), 18-180
IPASCII object class (table), 18-183
Junction object class (table), 18-190
$Keyboard global object (table), 18-192 to 18-193
LatchGate object class (table), 18-194
logical data members, 2-5
Maximum object class (table), 18-195
Minimum object class (table), 18-197
Mitsubishi and MitsubishiFX object classes (table), 18-201 to 18-202
ModbusSlave object class (table), 18-217 to 18-218
Multistate object class (table), 18-220
National Instruments Fieldbus (table), 18-223 to 18-224
National Instruments FieldPoint (table), 18-228 to 18-229
National Instruments Lookout OPC Client Driver (table), 18-234 to 18-235
native members, 2-4
Neutralzone object class (table), 18-241
NIDAQDevice object class (table), 18-243
NISCXI object class (table), 18-247 to 18-248
numeric data members, 2-6 to 2-10
Omron object class (table), 18-253
OneShot object class (table), 18-256
Optomux object class (table), 18-259 to 18-260
Pager object class (table), 18-265
Panel object class (table), 18-274 to 18-275
Pareto object class (table), 18-279 to 18-281
Philips object class (table), 18-284
PID object class (table), 18-290
Pipe object class (table), 18-292
Playwave object class (table), 18-293
Pot object (table), 18-296
Profibus DP object class (table), 18-299
ProfibusL2 object class, 18-306 to 18-307
Pulse object class (table), 18-310
Pushbutton object class (table), 18-313
Reliance object class (table), 18-332 to 18-333
RKC F Series object class (table), 18-316 to 18-322
Run object class (table), 18-334
S5_3964 object class (table), 18-337 to 18-340
Sample object class (table), 18-342
SampleText object class (table), 18-344
Scale object class (table), 18-346
SiemensTI505 (table), 18-350 to 18-354
Sixnet object class, 18-357 to 18-360
Spinner object class (table), 18-363
Spreadsheet object class (table), 18-367
SqlExec object class (table), 18-369
SquareD object class (table), 18-378 to 18-379
summing, in ASCII object, 18-80
Switch object class (table), 18-383
$System global object (table), 18-384
Tesco object class ``t, 18-386 to 18-388
text data members, 2-10 to 2-11
TextEntry object class (table), 18-391
TimeOf*xxxx* object class (table), 18-393
Tiway object class (table), 18-398 to 18-401
Toshiba Mseries/Toshiba Tseries (table), 18-405 to 18-407
Trend object class (table), 18-412
types of, 2-4
Wizdom object class (table), 18-413 to 18-414
XBarR object class (table), 18-418 to 18-420
XChart object class (table), 18-422 to 18-423

XYChart object class (table), 18-425 to 18-426
data transforms, 12-4 to 12-5
database service, 2-16
databases. *See* object databases.
DataTable object class, 18-90 to 18-102
connecting signals, 18-94 to 18-96
to cells, 18-94
to columns, 18-94 to 18-95
cursors, 18-98 to 18-100
data members (table), 18-100 to 18-102
Definition Parameters dialog box, 18-90 to 18-91
example, 18-92 to 18-97
connecting signals, 18-94 to 18-96
display panel, 18-96 to 18-97
reading cell value back to Lookout object, 18-95 to 18-96
multiple cursors, 18-99 to 18-100
multiplexing displays and graphics, 18-91 to 18-92
operating your multiplexed panel, 18-97 to 18-98
dates. *See also* time.
absolute dates and times (table), 2-10
date/time functions (table), 7-27
DDE (dynamic data exchange), 13-1 to 13-7
alarms, 13-6 to 13-7
client, 13-3 to 13-4
DDE service in Lookout, 2-17
linking Lookout to other applications, 13-2 to 13-5
networking considerations, 14-11 to 14-14
adding trusted DDE share, 14-13 to 14-14
running NETDDE.EXE automatically, 14-12
overview, 13-1 to 13-2
peer-to-peer, 13-4 to 13-5
server, 13-2 to 13-3
DdeLink object class, 18-103 to 18-104
data members (table), 18-104
Definition Parameters dialog box
remote computer, 18-103 to 18-104
same computer, 18-103
DdeTable object class, 18-105 to 18-108
data members (table), 18-108
Definition Parameters dialog box
remote computer, 18-107
same computer, 18-105 to 18-107
DelayOff object class, 18-109 to 18-110
data members (table), 18-110
Definition Parameters dialog box, 18-109 to 18-110
Display Parameters dialog box, 18-110
DelayOn object class, 18-111 to 18-112
data members (table), 18-112
Definition Parameters dialog box, 18-111 to 18-112
Display Parameters dialog box, 18-112
Delete command
Edit menu, 17-3
Object menu, 17-15 to 17-16
Run menu, 16-11
DeltaTau object class, 18-113 to 18-114
configuration parameters, 18-113
data members (table), 18-113 to 18-114
Derivative object class, 18-115 to 18-116
data members (table), 18-116
Definition Parameters dialog box, 18-115 to 18-116
Deselect All command, Alarms menu, 16-14
development environment, 3-6 to 3-7
process file, 3-6
source code file, 3-6 to 3-7
state file, 3-7
development process. *See also* process files.
example, 5-1 to 5-24
adding control objects, 5-11 to 5-16
adding data display, 5-9 to 5-10

completing interface panel, 5-17 to 5-20
connecting PLC and RTU, 5-20 to 5-23
creating control panel, 5-1 to 5-2
water level simulator, 5-3 to 5-9
overview, 3-7 to 3-8
DH+ interface parameters, AB object classes, 18-5 to 18-8
DialGauge object class, 18-117 to 18-119
data members, 18-119
Definition Parameters dialog box, 18-117 to 18-119
Display Parameters dialog box, 18-118
displaying dynamic graphics (table), 8-6
Dialing prefix settings (table), 6-6
Dial-up serial connection, 6-6
Display Options command, Alarms menu, 16-12
Displayable Object command, Insert menu, 17-4
DL205 and DL405 object classes
data members (table), 18-123 to 18-125
Definition Parameters dialog box, 18-121 to 18-123
status messages, 18-125 to 18-126
driver objects
purpose and use, 6-1 to 6-2
types of, 6-1
dynamic data exchange. *See* DDE (dynamic data exchange).
dynamic graphics, 8-6 to 8-13. *See also* graphics.
displaying
logical signals, 8-7 to 8-9
numeric signals, 8-9 to 8-12
text signals, 8-12 to 8-13
tools for displaying (table), 8-6
Dynamic object class, 18-127 to 18-132
configuration Parameters dialog box, 18-128 to 18-129
data members (table), 18-129 to 18-132

E

Edit commands, 17-1 to 17-3
Copy, 17-2
Cut, 17-2
Delete, 17-3
Edit Mode, 16-6, 17-3
Paste, 17-2
Select All, 17-3
Undo, 17-1 to 17-2
Edit Connections command, Object menu, 17-16
Edit Connections dialog box
$Alarm global object, 18-30
$Keyboard global object, 18-191
Edit Database command, Object menu, 17-16
Edit mode command, 16-6, 17-3
Edit mode menu commands, 17-1 to 17-21
Arrange commands, 17-16 to 17-18
Change functions, 17-19 to 17-21
Edit commands, 17-1 to 17-3
Insert commands, 17-4 to 17-10
Object commands, 17-11 to 17-16
editing database parameters, 4-5 to 4-13
logical parameters, 4-12 to 4-13
numeric parameters, 4-8 to 4-12
text parameters, 4-13
editing object parameters (example), 4-5 to 4-8
ElapsedTime object class
data members (table), 18-133
Definition Parameters dialog box, 18-133
electronic support services, A-1 to A-2
e-mail support, A-2
environmental services, 2-15 to 2-17
alarm service, 2-16
database service, 2-16
DDE service, 2-17
graphic service, 2-16

historical logging service, 2-17
multimedia service, 2-16
networking service, 2-17
ODBC service, 2-17
redundancy service, 2-17
security service, 2-16
serial port communication service, 2-15 to 2-16
error messages. *See also* status messages.
ASCII object class, 18-80 to 18-81
IPASCII object class, 18-188 to 18-189
NIDAQDevice object class, 18-244
NISCXI object class, 18-249
SquareD object class, 18-380
Ethernet interface parameters, AB object classes, 18-8
Event Logger, 11-9 to 11-10. *See also* logging data and events.
data location, 11-9
information overload, 11-10
Event object class
data members (table), 18-134
Definition Parameters dialog box, 18-134
event-driven processing, 2-13 to 2-15
advantages of active notification, 2-14 to 2-15
architecture (figure), 2-14
loop-driven applications compared with, 2-13
events, printing, 16-4
EXACT function, 7-22
Exit command, File menu, 16-5 to 16-6
EXP function, 7-17
exponential/scientific notation (table), 2-8
exporting object databases, 4-13 to 4-16
copying object databases, 4-19
creating database spreadsheet, 4-15 to 4-16
overview, 4-13
procedure, 4-13 to 4-15
Expression command, Insert menu, 17-5
Expression dialog box, 7-8 to 7-9
Expression editor dialog box, 3-10
expression fields
definition, 3-9
remembering tagnames, 3-9 to 3-10
expression functions, 7-13 to 7-27
date/time functions (table), 7-27
logical functions (table), 7-14 to 7-15
lookup functions (table), 7-16
mathematical functions (table), 7-17 to 7-19
statistical functions (table), 7-20 to 7-21
text functions (table), 7-22 to 7-24
trigonometric functions (table), 7-25 to 7-26
Expression object class, 18-135 to 18-136
Create Expression dialog box, 18-135
creating Expression objects, 7-4 to 7-5
example, 5-12 to 5-13
data members (table), 18-136
expressions, 7-1 to 7-27
as connections, 7-6 to 7-7
as parameters, 7-5 to 7-6
creating, 7-3 to 7-7
displaying on control panels, 7-3 to 7-4
examples, 7-1 to 7-2
functions, 7-13 to 7-27
illegal conditions, 7-2 to 7-3
inserting tagname into, 5-5
result of, 7-1
syntax, 7-9 to 7-13
arithmetic operators, 7-10
comparison operators, 7-11 to 7-13
text operator, 7-10
white space, 7-9
tools for displaying dynamic graphics (table), 8-6
yellow data entry fields (note), 7-3

F

FACT function, 7-17
FALSE function, 7-14
fax and telephone support numbers, A-2
Fax-on-Demand support, A-2
Fieldbus. *See* National Instruments Fieldbus.
FieldPoint object class. *See* National Instruments FieldPoint.
File commands, 16-1 to 16-6
  Close, 16-2
  Exit, 16-5 to 16-6
  Log off, 16-5
  Log on, 16-4 to 16-5
  New, 16-1
  Open, 16-1 to 16-2
  Password, 16-5
  Print, 16-4
  Reopen, 16-3
  Save, 16-3
Filter Options command, Alarms menu, 16-12 to 16-13
FIND function, 7-22
FisherROC object class, 18-137 to 18-143
  data members (table), 18-139 to 18-141
  Definition Parameters dialog box, 18-137 to 18-139
  status messages, 18-142 to 18-143
FIXED function, 7-22
Flipflop object class
  data members (table), 18-144
  Definition Parameters dialog box, 18-144
fonts, changing, 17-19
format strings. *See* ASCII object class; IPASCII object class.
fractional numbers with trailing zeroes (table), 2-8
FTP board support, A-1

G

Gauge object class, 18-145 to 18-146
  data members (table), 18-146
  Definition Parameters dialog box, 18-145 to 18-146
  Display Parameters dialog box, 18-146
  displaying dynamic graphics (table), 8-6
general numeric format (table), 2-7
GE_Series6 object class, 18-147 to 18-151
  data members (table), 18-149 to 18-150
  Definition Parameters dialog box, 18-147 to 18-149
  status messages, 18-150 to 18-151
GE_Series90 object class, 18-152 to 18-156
  data members (table), 18-153 to 18-155
  Definition Parameters dialog box, 18-152 to 18-153
  status messages, 18-155 to 18-156
global object classes
  $Alarm global object, 18-30 to 18-32
  created automatically by Lookout, 2-11
  definition, 2-11
  $Keyboard global object, 18-191 to 18-193
  $System global object, 18-384
Graphic command, Insert menu, 17-6 to 17-7
graphic file types
  bitmaps *vs.* metafiles, 8-17 to 8-18
  memory considerations, 8-18
graphics, 8-1 to 8-18
  animating. *See* Animator object class.
  creating custom graphics, 8-13 to 8-16
  dynamic graphics, 8-6 to 8-13
  file types, 8-17 to 8-18
  inserting into process file (example), 5-7 to 5-8
  Lookout graphics service, 2-16
  multiplexing displays and graphics, using DataTables, 18-91 to 18-92

screen resolution considerations (note), 8-1
static graphics, 8-1 to 8-5
Group command, Arrange menu, 17-18

H

Hand - Off - Auto modes, Alternator object class, 18-36 to 18-37
Hand - Off - Auto switch, adding to process file, 5-12 to 5-15
hardware key (note), 1-3
hardware networking, 14-10
hardware requirements, 1-1
Hardwired serial connections, 6-4
hexadecimal formats (table), 2-8 to 2-9
Histogram object class, 18-157 to 18-160
data members (table), 18-159 to 18-160
Definition Parameters dialog box, 18-157 to 18-159
historical database logging. *See* Citadel Historical Database Logger.
historical logging service, 2-17
Hitachi object class, 18-161 to 18-164
data members (table), 18-162 to 18-164
Definition Parameters dialog box, 18-161 to 18-162
status messages, 18-164
HOA. *See* Hand - Off - Auto.
HyperTrend object class, 18-165 to 18-172
button bar, 18-165
Cursor dialog box, 18-167
data members (table), 18-171 to 18-172
date and time indicators, 18-166
Definition Parameters dialog box, 18-168 to 18-171
Display Parameters dialog box, 18-171
use of Citadel database instead of state file (note), 3-7

I

(implicit) data members, 2-11
Import APT Database command, Options menu, 16-10
importing object databases
copying object databases, 4-19
overview, 4-13
procedure, 4-16 to 4-19
Insert commands, 17-4 to 17-10
Control Panel, 17-8 to 17-9
Displayable Object, 17-4
Expression, 17-5
Graphic, 17-6 to 17-7
Positions, 17-9 to 17-10
Scale, 17-7 to 17-8
Text/Plate/Inset, 17-5 to 17-6
insets
displaying in graphics, 8-1 to 8-3
inserting, 17-5 to 17-6
installation
automatic process loading, 1-5 to 1-6
hardware requirements, 1-1
procedure for, 1-2 to 1-3
software requirements, 1-1
starting Lookout for first time, 1-3 to 1-6
INT function, 7-17
Integral object class, 18-173 to 18-174
data members, 18-174
Definition Parameters dialog box, 18-173 to 18-174
Interpolate object class, 18-175 to 18-178
data members (table), 18-178
Definition Parameters dialog box, 18-175 to 18-177
Interval object class, 18-179 to 18-180
data members (table), 18-180
Definition Parameters dialog box, 18-179 to 18-180
Display Parameters dialog box, 18-180

IPASCII object class, 18-181 to 18-189. *See also* ASCII object class.
data members (table), 18-182 to 18-183
data types allowed (table), 18-185
Definition Parameters dialog box, 18-181 to 18-182
entering format string, 18-187
error messages, 18-188 to 18-189
markers, 18-184 to 18-187
request and response format strings, 18-183 to 18-184
request frame construction examples, 18-187 to 18-188
response construction examples, 18-187 to 18-188
Is equal to (=) operator, 7-11
Is greater than (>) operator, 7-11
Is greater than or equal to (>=) operator, 7-11
Is less than (<) operator, 7-11
Is less than or equal to (<=) operator, 7-11
Is not equal to (<>) operator, 7-11

J

Junction object class
data members (table), 18-190
Definition Parameters dialog box, 18-190
Justify Text command, 17-21

K

key, hardware (note), 1-3
keyboard, virtual. *See* virtual keyboard.
$Keyboard global object, 18-191 to 18-193
created automatically, 2-11
data members (table), 18-192 to 18-193
description, 18-191 to 18-192
Edit Connections dialog box, 18-191
keypad, virtual, 3-5
KT cards. *See* AB object classes.

L

LatchGate object class
data members, 18-194
Definition Parameters dialog box, 18-194
LCHOOSE function, 7-16
leading zeroes (table), 2-7
LEFT function, 7-22
LEN function, 7-23
LIF function, 7-14
lines, displaying in graphics, 8-1 to 8-3
LN function, 7-17
Log alarms to parameter, System command, 16-7
LOG function, 7-17
Log off command, File menu, 16-5
Log on command, File menu, 16-4 to 16-5
LOG10 function, 7-18
logging data and events, 11-1 to 11-12
Citadel Historical Database Logger, 11-5 to 11-9
Event Logger, 11-9 to 11-10
report generation, 11-10 to 11-12
Spreadsheet Logger, 11-1 to 11-5
logical data members
editing parameters, 4-12 to 4-13
purpose and use, 2-5
logical expressions
displaying dynamic graphics (table), 8-6
functions (table), 7-14 to 7-15
logical signals, displaying in graphics, 8-7 to 8-9
Lookout
architecture, 2-2 to 2-17
automatic process loading, 1-5 to 1-6
configuration shortcuts, 3-8 to 3-10
hardware requirements, 1-1
installation procedure, 1-2 to 1-3
National Instruments data acquisition devices supported (table), 18-244 to 18-245

overview, 2-1 to 2-2
registration (note), 1-3
SCXI devices supported (table), 18-250
software requirements, 1-1
starting, 1-3 to 1-6, 3-1 to 3-2
Lookout screen, 3-2 to 3-5
control panels, 3-4 to 3-5
illustration, 3-2
menu bar, 3-3
operator input, 3-5
status bar, 3-3
title bar, 3-2
workspace, 3-3
lookup functions (table), 7-16
loop-driven applications, compared with event-driven processing, 2-13
LOWER function, 7-23

M

mathematical functions (table), 7-17 to 7-19
MAX function, 7-20
Maximum object class, 18-195 to 18-196
data members (table), 18-195
Definition Parameters dialog box, 18-195
menu bars
overview, 3-3
viewing security, 10-7
Microsoft Access
accessing Citadel data, 12-10 to 12-11
compliance with ODBC (note), 12-3
Microsoft Excel, accessing Citadel data, 12-9 to 12-10
Microsoft Query
accessing Citadel data, 12-6 to 12-9
compliance with ODBC (note), 12-3
Microsoft Visual Basic
accessing Citadel data, 12-12
compliance with ODBC (note), 12-3
MID function, 7-23
MIN function, 7-20
Minimize All command, Windows menu, 16-14
Minimum object class, 18-197 to 18-198
data members (table), 18-197
Definition Parameters dialog box, 18-197
Mitsubishi and MitsubishiFX object classes, 18-199 to 18-203
Configuration Parameters dialog box, 18-199 to 18-201
data members (table), 18-201 to 18-202
Mitsubishi models supported, 18-203
status messages, 18-202 to 18-203
MOD function, 7-18
Modbus and ModbusMOSCAD object classes, 18-204 to 18-215
advanced parameters, 18-207 to 18-208
Configuration Parameters dialog box, 18-205 to 18-207
data members
6-digit address coding (table), 18-209 to 18-210
Modbus (table), 18-210 to 18-213
ModbusMOSCAD (table), 18-214 to 18-215
Modbus Protocol Statistics, 18-208 to 18-209
overview, 18-204
Modbus command, Options menu, 16-9
ModbusSlave object class, 18-216 to 18-218
Configuration Parameters dialog box, 18-217
data members (table), 18-217 to 18-218
overview, 18-216
modem settings, 6-6 to 6-7
Modify command, Object menu, 17-13 to 17-14
monitoring system development example. *See* development process.
More Windows command, Windows menu, 16-14
mouse shortcuts, 3-8 to 3-9

Move to Back command, Arrange menu, 17-18
Move to Front command, Arrange menu, 17-18
multilink networking, 14-2 to 14-5
compared with table networking, 14-11
linking controllable objects, 14-3 to 14-5
linking non-controllable objects, 14-5 to 14-6
multimedia service, 2-16
multiplexing displays and graphics, using DataTables, 18-91 to 18-92
Multistate object class, 18-219 to 18-220
Configuration Parameters dialog box, 18-219
data members (table), 18-220
displaying dynamic graphics (table), 8-6

N

National Instruments data acquisition devices supported by Lookout (table), 18-244 to 18-245
National Instruments Fieldbus, 18-221 to 18-225
alarms, 18-223
Configuration Parameters dialog box, 18-222 to 18-223
data members (table), 18-223 to 18-224
overview, 18-221
status messages, 18-224 to 18-225
troubleshooting, 18-225
National Instruments FieldPoint, 18-226 to 18-232
Configuration Parameters dialog box, 18-226 to 18-227
data members (table), 18-228
multiple discrete data members, 18-229
National Instruments Lookout OPC Client Driver, 18-233 to 18-239
adding single alias, 18-236 to 18-237
browsing server address space, 18-238
Configuration Parameters dialog box, 18-233 to 18-234
data members (table), 18-234 to 18-235
importing alias lists, 18-238 to 18-239
OPC item ID format, 18-236
OPC item IDs, 18-235
native data members, 2-4
NCHOOSE function, 7-16
networking, 14-1 to 14-14. *See also* redundancy.
capabilities, 14-1
DDE considerations, 14-11 to 14-14
adding trusted DDE share, 14-13 to 14-14
running NETDDE.EXE automatically, 14-12
hardware networking, 14-10
Lookout networking service, 2-17
methods, 14-2
multilink, 14-2 to 14-5
compared with table networking, 14-11
linking controllable objects, 14-3 to 14-5
linking non-controllable objects, 14-5 to 14-6
overview, 14-2
requirements, 14-1
table networking, 14-6 to 14-9
compared with multilink networking, 14-11
examples, 14-7 to 14-8
routing signals to and from driver objects, 14-8
topography (figure), 14-8
Neutralzone object class, 18-240 to 18-241
adding to process file, 5-12 to 5-14
data members (table), 18-241
Definition Parameters dialog box, 18-240
New command, File menu, 16-1
NIDAQDevice object class, 18-242 to 18-245

Configuration Parameters dialog box, 18-242 to 18-243
data members (table), 18-243
error messages, 18-244
National Instruments data acquisition devices supported (table), 18-244 to 18-245
NIDAQ.INI file, 18-243 to 18-244
NIDAQ.INI file configuration
NIDAQDevice object class, 18-243 to 18-244
NISCXI, 18-248 to 18-249
NIF function, 7-14
NISCXI object class, 18-246 to 18-250
configuring NIDAQ.INI for NISCXI, 18-248 to 18-249
channel attributes, 18-248
cold-junction sensor attributes, 18-249
data members (table), 18-247 to 18-248
Definition Parameters dialog box, 18-246 to 18-247
error messages, 18-249
SCXI devices supported (table), 18-250
Normal control panels, 3-4
NOT function, 7-14
NOW function, 7-27
nTitle command, Windows menu, 16-15
numeric constants, 2-6
numeric data members, 2-6 to 2-10
absolute dates and times (table), 2-10
definition, 2-6
editing parameters, 4-8 to 4-12
exponential/scientific notation (table), 2-8
fractional numbers with trailing zeroes (table), 2-8
general numeric format (table), 2-7
hexadecimal formats (table), 2-8 to 2-9
leading zeroes (table), 2-7
numeric constants, 2-6
time or time signals, 2-6 to 2-7
time period (table), 2-9 to 2-10
numeric expressions, for displaying dynamic graphics (table), 8-6
numeric format
changing, 17-20
general format (table), 2-7
numeric signals, displaying in graphics, 8-9 to 8-12

O

object classes. *See also* individual object classes, *e.g.*, *Animator object.*
alphabetical list, 18-1 to 18-426
definition, 2-11
global, 2-11
overview, 2-11
Object commands, 17-11 to 17-16
Create, 17-11 to 17-13
Delete, 17-15 to 17-16
Edit Connections, 17-16
Edit Database, 17-16
Modify, 17-13 to 17-14
object databases. *See also* data members.
copying, 4-19
creating database-generated alarms, 9-1 to 9-2
database service, 2-16
editing parameters, 4-5 to 4-13, 17-16
exporting, 4-13 to 4-16
importing, 4-16 to 4-19
overview, 2-4
object parameters. *See also* individual object classes, *e.g.*, *Animator object.*
connecting to data members, 4-19 to 4-20
definition, 2-3
editing, 4-5 to 4-13
example, 4-5 to 4-8
logical parameters, 4-12 to 4-13
numeric parameters, 4-8 to 4-12
text parameters, 4-13

expressions as parameters, 7-5 to 7-6
overview, 2-3
objects
connecting, 4-19 to 4-23
Alternator object class, 18-35
data members to data members, 4-21 to 4-23
data members to parameters, 4-19 to 4-20
editing connections, 17-16
example (figure), 2-12
expressions as connections, 7-6 to 7-7
identifying object data members, 4-20 to 4-21
overview, 2-12
PLC and RTU (development process example), 5-20 to 5-23
selecting target object (note), 4-19
signals, DataTable object class, 18-94 to 18-96
supervisory control, 2-12
copying and pasting (note), 4-4
creating, 4-2 to 4-5
Create command, 17-11 to 17-13
steps for, 4-2 to 4-4
tagnames, 4-4 to 4-5
definition, 2-2
deleting, 17-15 to 17-16
functionality, 2-3
modifying, 17-13 to 17-14
networking considerations
controllable objects, 14-3 to 14-5
non-controllable objects, 14-5 to 14-6
selecting, 4-1, 4-2
tagnames. *See* tagnames.
ODBC
definition, 12-1
Lookout ODBC service, 2-17
ODBC driver
accessing Citadel data, 12-1 to 12-12
data transforms, 12-4 to 12-5
ODBC-compliant applications (note), 12-3
SQL examples, 12-5 to 12-6
traces table, 12-3 to 12-4
with Microsoft Access, 12-10 to 12-11
with Microsoft Excel, 12-9 to 12-10
with Microsoft Query, 12-6 to 12-9
with Microsoft Visual Basic, 12-12
configuring, 12-1 to 12-3
Omron object class, 18-251 to 18-254
data members (table), 18-253
Definition Parameters dialog box, 18-251 to 18-252
models supported, 18-254
status messages, 18-254
OneShot object class, 18-255 to 18-256
data members (table), 18-256
Definition Parameters dialog box, 18-255 to 18-256
Display Parameters dialog box, 18-256
OPC Client Driver. *See* National Instruments Lookout OPC Client Driver.
Open command, File menu, 16-1 to 16-2
Open Database Connectivity (ODBC). *See* ODBC.
operator input, 3-5
operators
arithmetic, 7-10
comparison, 7-11 to 7-13
text, 7-10
Options commands
Accounts, 16-8 to 16-9
Import APT Database, 16-10
Modbus, 16-9
Serial Ports, 16-9 to 16-10
System, 16-6 to 16-8

Optomux object class, 18-257 to 18-262
data members (table), 18-259 to 18-260
Definition Parameters dialog box, 18-257 to 18-258
status messages, 18-261 to 18-262
watchdog capability, 18-259
OR function, 7-14

P

Pager object class, 18-263 to 18-266
Alphanumeric Definition Parameters dialog box, 18-263 to 18-264
data members (table), 18-265
Numeric Only Definition Parameters dialog box, 18-263 to 18-264
object modes
Alphanumeric, 18-265 to 18-266
Numeric Only, 18-265
queuing, 18-266
serial port settings, 18-265 to 18-266
status messages, 18-266
Panel object class, 18-267 to 18-275. *See also* control panels.
data members (table), 18-274 to 18-275
Definition Parameters dialog box, 18-267 to 18-270
"home panel" considerations, 18-272 to 18-273
manipulating panels, 18-270
printing, 18-273
screen resolution and graphics, 18-274
switching between panels, 18-270 to 18-272
parameters. *See* object parameters.
Pareto object class, 18-276 to 18-281
data members (table), 18-279 to 18-281
Definition Parameters dialog box, 18-276 to 18-277
Display Parameters dialog box, 18-277
incrementing factor counts, 18-279
weighted or unweighted charts, 18-278
passive notification system, 2-13
Password command, File menu, 16-5
passwords
forgetting your password (note), 10-3
protecting process files, 10-3 to 10-4
revising accounts, 10-3
Paste command, Edit menu, 17-2
Pause between calls (modem) setting, 6-7
peer-to-peer links, DDE, 13-4 to 13-5
Philips object class, 18-282 to 18-285
alarms, 18-285
Configuration Parameters dialog box, 18-282 to 18-283
data members (table), 18-284
status messages, 18-285
PI function, 7-18
PID object class, 18-286 to 18-291
data members (table), 18-290
Definition Parameters dialog box, 18-286 to 18-289
positional control, 18-289
velocity control, 18-289 to 18-290
Pipe object class
data members (table), 18-292
Definition Parameters dialog box, 18-292
displaying dynamic graphics (table), 8-6
plates
displaying in graphics, 8-1 to 8-3
inserting, 5-17 to 5-18, 17-5 to 17-6
Playwave object class
data members (table), 18-293
Definition Parameters dialog box, 18-293
PLCs. *See also* AB object classes.
connecting to Lookout (example), 5-20 to 5-23
Poll Rate, 6-2
Popup control panels
definition, 3-4
parameter settings, 18-267 to 18-268

viewing security, 10-7
Popup no icon control panels, 18-268
Position command, Insert menu, 17-9 to 17-10
Pot object class, 18-294 to 18-296
data members (table), 18-296
Definition Parameters dialog box, 18-294 to 18-295
development example, 5-3 to 5-5
Display Parameters dialog box, 18-295
displaying dynamic graphics (table), 8-6
displaying dynamic numeric signals, 8-9 to 8-12
generation of numeric signals, 2-6
Pot Display dialog box (figure), 4-4
repositioning using sizing handles, 5-4 to 5-5
selecting display type (example), 5-4
power failure, automatic process loading after, 1-5 to 1-6
Print command
Alarms menu, 16-13
File menu
Alarms option, 16-4
Events option, 16-4
printing
alarms, 9-10
panels, 18-273
prioritizing alarms, 9-4
process files. *See also* development process.
definition, 3-6
inserting graphics, 5-7 to 5-8
loading automatically, 1-5 to 1-6
opening, 3-2
protecting, 10-3 to 10-4
purpose and use, 3-6
PRODUCT function, 7-18
Profibus DP object class, 18-297 to 18-300
Configuration Parameters dialog box, 18-298
configuring DP network, 18-297
data members (table), 18-299
PFB card settings, 18-298 to 18-299
requirements for running, 18-297
status messages, 18-299 to 18-300
ProfibusL2 object class, 18-301 to 18-308
Configuration Parameters dialog box, 18-304 to 18-305
data members, 18-306 to 18-307
Lookout messaging system, 18-301
PFB card settings, 18-305 to 18-306
requirements, 18-304 to 18-306
sample program, 18-301 to 18-302
DB1 configuration, 18-302 to 18-303
detailed explanation, 18-302 to 18-304
function block explanation, 18-303 to 18-304
status messages, 18-307 to 18-308
PROPER function, 7-23
Pulse object class, 18-309 to 18-310
data members (table), 18-310
Definition Parameters dialog box, 18-309 to 18-310
Display Parameters dialog box, 18-310
Pushbutton object class, 18-311 to 18-313
action verification, 10-8
data members (table), 18-313
Definition Parameters dialog box, 18-311 to 18-313
Display Parameters dialog box, 18-313
displaying dynamic graphics (table), 8-6
Verification Message dialog box, 18-312

R

Radio (RTS/CTS) serial connection, 6-4
RAND function, 7-18
Receive gap setting, 6-4
Recipe object class, 18-324 to 18-330
data members (table), 18-329
Definition Parameters dialog box, 18-325 to 18-326

Display Parameters dialog box, 18-328
example, 18-324 to 18-325
file selection dialog box (figure), 18-326 to 18-327
omitting ingredients (note), 18-329
reloading files for changed recipe (note), 18-327
rectangles, displaying in graphics, 8-1 to 8-3
redundancy, 15-1 to 15-10
basic standby principles, 15-2 to 15-3
defining network settings, 15-4 to 15-5
enabling file sharing
Windows 3.1.1, 15-5 to 15-6
Windows 95/NT, 15-7 to 15-8
failover occurrence (note), 15-2
failover scenarios, 15-3
Lookout redundancy service, 2-17
option configuration, 15-8 to 15-10
standby configuration, 15-3 to 15-10
registration of Lookout
changing data, 16-8
hardware key (note), 1-3
permanent unlock code (note), 1-3
Reliance object class, 18-331 to 18-333
data members (table), 18-332 to 18-333
Definition Parameters dialog box, 18-331 to 18-332
destination settings, 18-332
PC-Link card settings, 18-332
status messages, 18-333
Reopen command, File menu, 16-3
REPLACE function, 7-23
report generation, 11-10 to 11-12
control panel reports, 11-10 to 11-12
third party reports, 11-12
REPT function, 7-23
request and response format strings
ASCII object class, 18-75 to 18-79
entering format string, 18-78
object markers, 18-76 to 18-78
request frame construction examples, 18-79
response format examples, 18-79
IPASCII object class, 18-183 to 18-184
entering format string, 18-187
markers, 18-184 to 18-187
request frame construction examples, 18-187
response format examples, 18-187 to 18-188
Retries (modem) setting, 6-7
Revise command, Run menu, 16-11
RIGHT function, 7-24
RKC F Series object class, 18-314 to 18-323
data members (tables), 18-316 to 18-322
Lookout data members, 18-322
measured value group, 18-317
memory area and set value group, 18-318
operation mode group, 18-317
parameter group 10 (measured input parameters), 18-318
parameter group 11 (remote setting input parameters), 18-318
parameter group 12 (output parameters), 18-319
parameter group 13 (auto-tuning bias parameters), 18-319
parameter group 14 (alarm 1 parameters), 18-319
parameter group 15 (analog output parameters), 18-320
parameter group 16 (positioning, proportioning, PID action parameters), 18-320
parameter group 17 (bar graph parameter), 18-320
parameter group 20 (input selection parameters), 18-320 to 18-321
parameter group 21 (setting parameters), 18-321

parameter group 22 (output action parameters), 18-321
parameter group 23 (alarm 2 parameters), 18-321 to 18-322
parameter group 40 (data lock parameters), 18-322
RKC data member group, 18-316
Definition Parameters dialog box, 18-314 to 18-315
RKC models supported, 18-323
status messages, 18-322 to 18-323
ROUND function, 7-18
RTS delay off time period, 6-5
RTS/CTS handshaking settings, 6-4 to 6-5
RTU Configuration dialog box, Aquatrol object class, 18-68 to 18-69
RTUs, connecting to Lookout (example), 5-20 to 5-23
Run commands, 16-10 to 16-11
Add, 16-10 to 16-11
Delete, 16-11
Revise, 16-11
Run object class, 18-334 to 18-335
data members (table), 18-334
Definition Parameters dialog box, 18-334
Runtime menu commands, 16-1 to 16-15
Alarm commands, 16-12 to 16-14
Edit command, 16-6
File commands, 16-1 to 16-6
Option commands, 16-6 to 16-10
Run commands, 16-10 to 16-11
Window commands, 16-14 to 16-15

S

S5_3964 object class, 18-336 to 18-341
alarms, 18-340 to 18-341
data members (table), 18-337 to 18-340
Definition Parameters dialog box, 18-336 to 18-337
Sample object class, 18-32 to 18-343
data members (table), 18-342
Definition Parameters dialog box, 18-342
SampleText object class
data members (table), 18-344
Definition Parameters dialog box, 18-344
Save command, File menu, 16-3
Scale command, Insert menu, 17-7 to 17-8
Scale object class, 18-345 to 18-346
adding scales to control panel, 5-19
data members (table), 18-346
Definition Parameters dialog box, 18-345
Display Parameters dialog box, 18-346
scientific notation (table), 2-8
screen resolution
display panels (note), 5-2
graphics (note), 8-1
SCXI devices supported (table), 18-250. *See also* NISCXI object class.
SEARCH function, 7-24
security, 10-1 to 10-8
accounts, 10-1 to 10-3
action verification, 10-8
control security, 10-4 to 10-5
Lookout security service, 2-16
overview, 10-1
Panel object class, 18-268
process file security, 10-3 to 10-4
setting with System command, 16-7
viewing security, 10-5 to 10-7
control panels, 10-5
controllable objects, 10-6
Panel object class, 18-268
system settings, 10-6 to 10-7
security levels, assigning to accounts, 10-2 to 10-3
Select All command
Alarms menu, 16-14
Edit menu, 17-3
Select graphic dialog box, Animator object class, 18-39
Select Object Class list box (figure), 4-2

selecting objects, 4-1
serial communications, 6-1 to 6-7
configuring for process (example), 5-20
defining serial port settings, 6-2 to 6-7
dial-up modem settings, 6-6 to 6-7
driver objects, 6-1 to 6-2
Hardwired connections, 6-4
Lookout serial port communication service, 2-15 to 2-16
overview, 6-2
Receive gap setting, 6-4
RTS/CTS handshaking settings, 6-4 to 6-5
selecting serial port, 6-3
Serial port data field, 6-3
serial port interface parameters
AB object class, 18-4
SquareD object class, 18-375
Serial Port Settings dialog box (figure), 6-3
Serial Ports command, Options menu, 16-9 to 16-10
server, DDE, 13-2 to 13-3
Show command, Alarms menu, 16-12
Siemens Simatic S5 PLCs. *See* S5_3964 object class.
SiemensTI505 object class, 18-347 to 18-355
configuring HI-TF, 18-348 to 18-350
data members (table), 18-350 to 18-354
Definition Parameters dialog box, 18-347 to 18-348
status messages, 18-354 to 18-355
SIGN function, 7-19
SIN function, 7-26
Sixnet object class, 18-356 to 18-361
Configuration Parameters dialog box, 18-356 to 18-357
data members, 18-357 to 18-360
importing Sixtags database, 18-360
status messages, 18-360 to 18-361
sizing handles, for repositioning objects, 5-4 to 5-5
sliders, adding to control panel, 5-18
software requirements, 1-1
source code file
definition, 3-6
purpose and use, 3-6 to 3-7
space, in expressions, 7-9
Space Evenly command, Arrange menu, 17-17 to 17-18
Spinner object class, 18-362 to 18-363
data members (table), 18-363
Definition Parameters dialog box, 18-362
displaying dynamic graphics (table), 8-6
Spreadsheet Logger, 11-1 to 11-5
concurrent file access, 11-4 to 11-5
CSV files, 11-3 to 11-4
data location, 11-2 to 11-3
file and disk errors, 11-3 to 11-4
information overload, 11-5
Spreadsheet object class, 18-364 to 18-367
Configuration Parameters dialog box, 18-364 to 18-367
data members (table), 18-367
exporting object database to spreadsheet file, 4-15 to 4-16
SQL. *See also* ODBC driver.
definition, 12-1
example queries, 12-1
SqlExec object class, 18-368 to 18-373
comma separated value (CSV) file support, 18-38-
comments, 18-369 to 18-371
Configuration Parameters dialog box, 18-368
data members (table), 18-369
sample DNS and SQL strings, 18-370 to 18-371
SQL command buffering, 18-371 to 18-372
status messages, 18-372 to 18-373
SQRT function, 7-19
SquareD object class, 18-374 to 18-380

data members (table), 18-378 to 18-379
Definition Parameters dialog box, 18-374 to 18-375
error messages, 18-380
serial port interface parameters, 18-375
SY/ENET interface parameters, 18-377 to 18-378
SY/LINK interface parameters, 18-375 to 18-376
S-S 5136-SD card, 18-9
standby
basic principles, 15-2 to 15-3
configuration, 15-3 to 15-10
defining network settings, 15-4 to 15-5
enabling file sharing
Windows 3.1.1, 15-5 to 15-6
Windows 95/NT, 15-7 to 15-8
failover occurrence (note), 15-2
failover scenarios, 15-3
option configuration, 15-8 to 15-10
Standby button, 16-8
starting Lookout, 1-3 to 1-6, 3-1 to 3-2
Startup process file parameter, System command, 16-6
state file
definition, 3-7
not used by Hypertrends (note), 3-7
purpose and use, 3-7
setting save option, 16-7
static graphics, 8-1 to 8-5
custom graphics, 8-3 to 8-5
displaying text, plates, insets, rectangles, and lines, 8-1 to 8-3
statistical functions (table), 7-20 to 7-21
status bar, 3-3
status messages. *See also* error messages.
Alternator object class, 18-37 to 18-38
Applicom object classes, 18-65 to 18-66
Aquatrol object class, 18-70 to 18-71
Cutler-Hammer object class, 18-88 to 18-89
DL205 and DL405 object classes, 18-125 to 18-126
FisherROC object class, 18-142 to 18-143
GE_Series6 object class, 18-150 to 18-151
GE_Series90 object class, 18-155 to 18-156
Hitachi object class, 18-164
Mitsubishi and MitsubishiFX object classes, 18-202 to 18-203
National Instruments Fieldbus, 18-224 to 18-225
Omron object class, 18-254
Optomux object class, 18-261 to 18-262
Pager object class, 18-266
Philips object class, 18-285
Profibus DP object class, 18-299 to 18-300
ProfibusL2 object class, 18-307 to 18-308
Reliance object class, 18-333
RKC F Series object class, 18-322 to 18-323
SiemensTI505, 18-354 to 18-355
Sixnet object class, 18-360 to 18-361
SqlExec object class, 18-372 to 18-373
Toshiba Mseries/Toshiba Tseries, 18-407
STDEV function, 7-20
STDEVP function, 7-20
Structured Query Language. *See* SQL.
sum data members, ASCII object, 18-80
SUM function, 7-21
Switch Definition dialog box (figure), 2-4
Switch object class, 18-381 to 18-383
action verification, 10-8, 18-321
data members (table), 2-5, 18-383
Definition Parameters dialog box, 18-381 to 18-382
displaying
dynamic graphics (table), 8-6
logical signals in graphics, 8-7 to 8-9

graphics for switches, 18-382
overview, 2-4
Verification Message dialog box, 18-381
System command, Options menu, 16-6 to 16-8
$System global object
created automatically, 2-11
data members (table), 18-384
description, 18-384
system options, setting for security, 10-6 to 10-7

T

table networking, 14-6 to 14-9
compared with multilink networking, 14-11
examples, 14-7 to 14-8
routing signals to and from driver objects, 14-8
topography (figure), 14-8
tabs, in expressions, 7-9
tagnames
characters allowed in tagnames, 4-4
definition, 4-4
inserting into expression field, 5-5
invalid names (example), 4-5
remembering, 3-9 to 3-10
valid names (example), 4-4
TAN function, 7-26
TCHOOSE function, 7-16
technical support, A-1 to A-2
telephone and fax support numbers, A-2
Tesco object class, 18-385 to 18-388
Configuration Parameters dialog box, 18-385 to 18-386
data members (table), 18-386 to 18-388
text
changing color, 17-19 to 17-20
displaying in graphics, 8-1 to 8-3
justifying, 17-21
text data members
editing, 4-13
examples of text constants, 2-11
purpose and use, 2-10
text expressions, for displaying dynamic graphics (table), 8-6
TEXT function, 7-24
text functions (table), 7-22 to 7-24
text operators, 7-10
text signals, displaying in graphics, 8-12 to 8-13
TextEntry object class, 18-389 to 18-391
data members (table), 18-391
Display Parameters dialog box, 18-390
Parameters dialog box, 18-389 to 18-390
Text/Plate/Inset command, Insert menu, 17-5 to 17-6
third party reports, 11-12
TIF function, 7-15
time
absolute dates and times (table), 2-10
date/time functions (table), 7-27
stored as numeric values, 2-6
time period constants (examples), 2-6 to 2-7
time periods (table), 2-9 to 2-10
Time zone option, System command, 16-8
TimeOf*xxxx* object class, 18-392 to 18-393
data members (table), 18-393
Definition Parameters dialog box, 18-392
Display Parameters dialog box, 18-393
title bars
overview, 3-2
viewing security, 10-7
Tiway object class, 18-394 to 18-402
communication techniques, 18-395 to 18-398
CTI TCP/IP, 18-398
local port, 18-395
Unilink Host Adapter, 18-395 to 18-396

Unilink PC Adapter, 18-396 to 18-397
Configuration Parameters dialog box, 18-394 to 18-395
data members (table), 18-398 to 18-401
importing APT tag files, 18-402
TODAY function, 7-27
Toshiba Mseries/Toshiba Tseries, 18-403 to 18-407
Configuration Parameters dialog box, 18-403 to 18-405
data members (table), 18-405 to 18-407
status messages, 18-407
traces table, for Citadel data, 12-3 to 12-4
trailing zeroes (table), 2-8
Transparent pixel data fields, 8-4
Trend object class, 18-408 to 18-412
data members (table), 18-412
Definition Parameters dialog box, 18-408 to 18-411
Display Parameters dialog box, 18-411
trigonometric functions (table), 7-25 to 7-26
TRIM function, 7-24
TRUE function, 7-15
TRUNC function, 7-19
trusted DDE share, 14-13 to 14-14

U

Undo command, Edit menu, 17-1 to 17-2
Ungroup command, Arrange menu, 17-18
unlock code for Lookout (note), 1-3
UPPER function, 7-24

V

VAR function, 7-21
VARP function, 7-21
viewing security, 10-5 to 10-7
control panels, 10-5, 18-268 to 18-269
controllable objects, 10-6
system settings, 10-6 to 10-7
virtual keyboard
enabling, 3-5
security options, 10-1
setting Pops Up On parameter, 16-7
virtual keypad, 3-5
Visual Basic
accessing Citadel data, 12-12
compliance with ODBC (note), 12-3

W

Wait for connection (modem) setting, 6-7
white space, in expressions, 7-9
Window commands, 16-14 to 16-15
Arrange Icons, 16-14
Minimize All, 16-14
More Windows, 16-15
nTitle, 16-15
Windows Metafile (.WMF) graphics
compared with bitmap files, 8-17 to 8-18
displaying, 8-3 to 8-5
Wizdom object class, 18-413 to 18-414
configuring, 18-413
data members (table), 18-413 to 18-414
.WMF. *See* Windows Metafile (.WMF) graphics.
workspace, 3-3

X

XBarR object class, 18-415 to 18-420
data members (table), 18-418 to 18-420
Definition Parameters dialog box, 18-415 to 18-417
Display Parameters dialog box, 18-416
R chart (figure), 18-417
X-bar chart (figure), 18-417
XChart object class, 18-421 to 18-423
data members (table), 18-422 to 18-423

Definition Parameters dialog box, 18-421 to 18-422
Display Parameters dialog box, 18-422
XOR function, 7-15
XYChart object class, 18-424 to 18-426
data members (table), 18-425 to 18-426
Definition Parameters dialog box, 18-424 to 18-425
Display Parameters dialog box, 18-425

Z

zeroes
fractional numbers with trailing zeroes (table), 2-8
leading zeroes (table), 2-7